2009

Analysis of virtual environments through a web based visualization tool

Ronald Valente

ROCHESTER INSTITUTE OF TECHNOLOGY

Analysis of virtual environments through a web based visualization tool

by

Ronald R. Valente

September 2009

Thesis submitted in partial fulfillment of the requirements for the
degree of Master of Science in
Networking and Systems Administration

B. Thomas Golisano College of Computing and Information Sciences
of
Department of Networking, Security, and Systems Administration

ROCHESTER INSTITUTE OF TECHNOLOGY

B. Thomas Golisano College of
B. Thomas Golisano College of Computing and Information Sciences

Master of Science in
Department of Networking, Security, and Systems Administration

Thesis Approval Form

Student Name: _____

Thesis Title: _____

Thesis Committee

Name	Signature	Date

_____

Committee Chair: Dr. Sumita Mishra

_____

Committee Member: Dr. Yin Pan

_____

Committee Member: Dr. Luther Troell

Declaration of Authorship

I, Ronald R. Valente, declare that this thesis titled, 'Analysis of virtual environments through a web based visualization tool' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

ROCHESTER INSTITUTE OF TECHNOLOGY

Abstract

B. Thomas Golisano College of Computing and Information Sciences
Department of Networking, Security, and Systems Administration

Masters of Networking and System Administration

by Ronald R. Valente
September 2009

Virtualization is gaining popularity in corporations as well as academia. This technology provides a large return on investment by providing the ability to consolidate hardware servers into smaller, more efficient virtual machines (VM). With a seemingly negligible cost to deploy a new virtual machine when compared to deploying a physical server, more and more virtual machines get deployed. However, as the number of virtual machine increases, the task of managing and keeping track of them becomes exceedingly cumbersome. With advanced visualization techniques and real-time analysis of virtualization environments, the control over virtual machine sprawl will become more manageable and thus improving virtual environment efficiency.

This thesis describes the development of a novel web-based visualization tool for the virtual environment. With this tool, it is possible to better monitor the environment and control the VM sprawl. This tool provides the potential to optimize the environment and maintain an optimal level of performance for each virtual machine in the environment. The details of the visualization tool and analysis of the obtained results are also provided.

Acknowledgements

I would like to thank the following people who have helped me not only through this thesis but my educational career...

My family and friends have been a continued source of support.

Dr. Sumita Mishra has been a great influence whose energy and enthusiasm throughout this entire project helped me stick with it.

Dr. Luther Troell for the guidancehe provided my throughout my undergraduate and graduate career at RIT.

Dr. Yin Pan who had a fantastic way of presenting material during courses which made everything more interesting than it should have been.

Contents

Declaration of Authorship ii

Abstract iii

Acknowledgements iv

List of Figures viii

List of Tables ix

1 Introduction 1
- 1.1 Virtualization Background . . . 1
  - 1.1.1 Benefits of Virtualization . . . 1
  - 1.1.2 Challenges of Virtualization . . . 2
- 1.2 Need for Tools and Analysis . . . 2
- 1.3 Work Presented . . . 3

2 Literature Review 4
- 2.1 Overview . . . 4
- 2.2 Virtualization . . . 4
- 2.3 Storage . . . 6
- 2.4 Visualization . . . 7
- 2.5 Conclusion . . . 8

3 Methodologies 9
- 3.1 Overview . . . 9
- 3.2 Environment . . . 9
- 3.3 Purpose . . . 11
- 3.4 Limitations . . . 11
  - 3.4.1 Assumptions . . . 12
  - 3.4.2 Validity . . . 12
  - 3.4.3 Growth . . . 13
- 3.5 Procedure . . . 13
  - 3.5.1 Data Collection . . . 14
  - 3.5.2 Visualization . . . 14
- 3.6 Conclusion . . . 15

4 Web Application 16
- 4.1 Overview . . . 16
  - 4.1.1 Database . . . 17
  - 4.1.2 Interface . . . 18
  - 4.1.3 Dashboard . . . 18
    - 4.1.3.1 Available Counters . . . 18
    - 4.1.3.2 Host Summary . . . 20
    - 4.1.3.3 Array Summary . . . 21

4.1.4 Navigation . . . 21
4.1.5 Hosts View . . . 23
4.1.6 Storage Array View . . . 25
4.2 Analysis . . . 26

5 Results & Discussion **28**
5.1 Overview . . . 28
5.2 Host Analysis . . . 28
5.2.1 CPU Usage . . . 29
5.2.2 Memory Usage . . . 32
5.2.3 Disk Usage . . . 35
5.2.4 Counter Overview . . . 36
5.2.5 Host Data Source Correlations . . . 37
5.3 Storage Analysis . . . 38
5.3.1 Utilization . . . 39
5.3.2 Queue Length . . . 40
5.3.3 Response Time . . . 41
5.3.4 Throughput . . . 42
5.3.5 Bivariate Throughput . . . 43
5.3.6 Bivariate Queue Length . . . 43
5.3.7 LUN Performance . . . 45
5.3.8 Disk Performance . . . 46
5.4 Conclusion . . . 47

6 Conclusion **48**
6.1 Overview . . . 48
6.2 Web Application . . . 48
6.3 Host Analysis . . . 49
6.3.1 Access Reduction . . . 50
6.3.2 Monitoring . . . 51
6.4 Storage Analysis . . . 51

7 Recommendations **52**
7.1 Future Work . . . 52
7.1.1 Collection Improvements . . . 52
7.1.2 Available Counters . . . 53
7.1.3 Realtime Analysis . . . 53
7.1.4 Flexible Visualization Code . . . 53
7.1.5 Pure Ruby Implementation . . . 53
7.1.6 Notifications . . . 54
7.2 Conclusion . . . 54

A Key Terms **55**
A.1 Virtualization . . . 55
A.2 Programming . . . 55

B Automation **56**
B.1 Capistrano . . . 56

C Host Data Collection **57**
C.1 ESX Host 57
C.2 esxtop configuration 58

D Data Summarization **59**
D.1 Host Summarization 59
D.2 Array Summarization 61

E ESX Host Data Collection **63**
E.1 VMware API Perl Script 63

F Web Application Database Schema **68**
F.1 Schema Design 68

G JMP Scripts **70**
G.1 LUN Bubble Plot 70

H Storage Array Collection Scripts **71**
H.1 Storage Data Collection Script 71

Bibliography **76**

List of Figures

2.1 MRTG Graphing Output[1] . . . 7
2.2 RRDtool Graphing Output[2] . . . 8

3.1 VMware Cluster Rack Layout . . . 10
3.2 Cluster and Storage Rack Layouts . . . 10

4.1 veViz Dashboard . . . 19
4.2 veViz Host Summary Detail . . . 20
4.3 veViz Array Summary Detail . . . 21
4.4 veViz Navigation Menu . . . 21
4.5 veViz Host Selection . . . 23
4.6 veViz Host Detail . . . 23
4.7 veViz Host Legend . . . 24
4.8 veViz Host Memory Usage . . . 24
4.9 veViz Array Selection . . . 25
4.10 veViz Array Detail . . . 26
4.11 veViz Array Legend . . . 26

5.1 CPU Usage (MHz) . . . 29
5.2 Host CPU Usage by Host . . . 30
5.3 Host CPU Usage Range . . . 31
5.4 Host CPU Time Series . . . 31
5.5 Memory Usage (%) . . . 32
5.6 Host Memory Time Series . . . 33
5.7 Overall Memory Usage . . . 34
5.8 Disk Usage (KB/sec) . . . 35
5.9 Counters vs. Timestamp by CPU Level . . . 36
5.10 CPU Memory and Disk Scatterplot Matrix . . . 37
5.11 Service Processor Distributions . . . 39
5.12 Bivariate: Utilization & Total Throughput . . . 43
5.13 Bivariate: Total Throughput & Queue Length . . . 44
5.14 Bivariate: Response Time & Queue Length . . . 44
5.15 Bivariate: Flush Ratio & Queue Length . . . 45
5.16 LUN Queue Length, Throughput, and Response Time . . . 46
5.17 Individual Disk Bubble Plot . . . 47

List of Tables

3.1 Virtual Environment Components 10
3.2 Virtual Environment Hardware Specifications 11

4.1 Available Environment Counters 19

5.1 Host CPU Counter Moments 29
5.2 Host CPU Level Color Legend 30
5.3 Host Memory Counter Moments 32
5.4 Host Memory Level Color Legend 34
5.5 Host Disk Counter Moments 35
5.6 Storage Array Objects 38
5.7 SP A/B Array Utilization Moments 40
5.8 SP A/B Array Utilization Quantiles 40
5.9 SP A/B Array Queue Length Moments 40
5.10 SP A/B Array Queue Length Quantiles 41
5.11 SP A/B Array Response Time Moments 41
5.12 SP A/B Array Response Time Quantiles 41
5.13 SP A/B Array Throughput Moments 42
5.14 SP A/B Array Throughput Quantiles 42

7.1 Data Rollup Policy 53

F.1 veViz Summary Schema 68
F.2 veViz Master Schema 69

1 Introduction

1.1 Virtualization Background

Virtualization has been around since the early 1980s. Virtualization was not extremely efficient or practical back when it was conceived. It took a large amount of time and effort to instantiate a single virtual machine. There was not a demand for virtualization because of the learning curve and additional steps required to utilize it. All that has changed substantially since then and now virtualization has made a surge into both the corporate and academic realms.

1.1.1 Benefits of Virtualization

By leveraging virtualization, one has the capability to isolate an operating system instance without requiring dedicated hardware for each instance. Having this flexibility provides the user with the ability to increase utilization of each hardware machine, thus allowing each hardware server to run at a higher level of utilization resulting in lower amount of servers needed overall. Ultimately this reduction in number of servers should be able to create a more cost effective and easier to maintain datacenter.

1.1.2 Challenges of Virtualization

With all of the positive effect that virtualization has made on corporate entities and academia, it is hard to even notice the drawbacks at times. For instance with large virtual environments, the ability to create and deploy a virtual machine is easy but keeping track of all the virtual instances can be challenging to say the least. Keeping track of these virtual machines is so challenging that there is an outbreak of virtual machines in virtual environments. This outbreak can be dedicated entirely to the difficulty to monitor and manage the purpose of virtual machines and the usage. It is not as simple as checking if the virtual machine is on or off. This problem ripples across the entire virtual environment because the number of virtual machines directly correlates with the performance of the environment as a whole.

1.2 Need for Tools and Analysis

Virtualization has been such a driving force in the information technology field, the ability to control the virtual environment is becoming harder as the environments grow in size and capability. There needs to be a revolutionary method for the analysis of a virtual environment which will permit a deeper look into the environment for better optimization and efficiency. Considering all of the variables that contribute to the overall performance of a virtual environment it would only make sense to run a multivariate analysis of the environment to determine interactions, if any. If any interactions are found, one may find the potential to improve performance without additional hardware just by optimizing variables within the bounds of the given environment.

Virtualization and the act of abstracting an application or operating system from the hardware is not a new idea by any stretch of the imagination. That being said using virtualization to improve system utilization, reduce power consumption, and increasing the flexibility and redundancy of the environment are just some of the reasons why its popularity is increasing so rapidly. No one solution is perfect, and leveraging virtualization has its caveats. Most importantly is the complexity of monitoring the system. There are now multiple layers of performance that must be monitored. On top of that storage is now centralized which can now be a bottleneck in some operations. The ability to flexibly and efficiently monitor these virtual environments is extremely important to the success of their implementation.

1.3 Work Presented

This thesis uses combination of programming, system administration, and statistical analysis to provide a set of methods to perform in-depth analysis of virtual environments. Data is collected using an optimized polling mechanism and save all the information into a relational database. Once the data is stored into a database further analysis can is performed via JMP, a statistical package. Statistical analysis of virtual machines and storage utilization using control charts and other statistical methods will is implemented. Each of these methods will is represented in a coherent manner to increase the readability to the end-user.

2

Literature Review

2.1 Overview

There are three overarching topics that have a role in virtual machine sprawl, virtualization, storage, and efficiency. By monitoring the virtualization and storage aspects of a virtual environment then and only then can virtual machine sprawl be tracked. When looking into a typical virtual environment deployment, it is important to state that there usually is not one admin. There are multiple admins in charge of different aspects of the system. For example, one admin would handle the storage, while another admin would monitor and control the ESX servers. This segmentation can create a disconnect when it comes to overall efficiency between the virtualization/host side and the storage area network side on the environment. With proper analysis and visualization it is possible to close the gap between the two main aspects of a virtual environment and get a better handle on virtual machine sprawl.

2.2 Virtualization

Virtualization has been an extremely popular topic lately due to its many touted advantages with seemingly minimal disadvantages. Virtualization has been made out to be such a great technology that any disadvantaged is out-weighed by the massive amount of advantages. These miniscule disadvantages in an environment that is in its infancy can evolve into severe problems in the future. The major disadvantages for a virtual

environment is directly related to one of its advantages, deployment. With a virtual environment, deploying the server consists of right clicking and selecting the new virtual machine operation. After completing the wizard and a few minutes go by, you have a brand new server ready to work on. The speed and ease of deployment has made the cost of server deployment disappear, at least to the end-user or specific server admin. The virtual machine admin on the other hand knows there is a specific cost to every VM and each cost is different based on the resource utilization of that VM.

With regards to the ease of deployment there is one major result, the over abundance of VMs in a virtual environment. This excessive creation of VMs creates a set of issues in the environment that may not have been an issue without the use of VMs. By having so many VMs running, manageability decreases, performance of each VM decreases, and storage availability decreases. The term for this proliferation of virtual machines in a virtual environment is called VM Sprawl.

There has been a few technologies that have been developed to slow the reduction in resources by creating new VMs in a virtual environment. Some of these technologies that have been developed cost more money or require more administration overhead to implement. VMware has introduced a technology in VMware Workstation[2] which leveraged the copy on write method to implement what they call *linked clones*. A group of engineers from IBM proposed a solution for VM sprawl which involved converting the current VM format to a proprietary format which increase the ability to compress and share data between VMs to lower storage utilization [3]. This solves one aspect of the VM sprawl issue but fails to handle resource utilization creep.

By leveraging the visualization of the virtual environment as opposed to using a proprietary VM disk format it should be possible to implement this solution without a change up-front to the environment. The visualization of the entire environment will give all users of the environment more insight on the usage of each component and not just the pieces they user every-day.

[2]VMware Workstation is VMware's premier desktop virtualization product.

2.3 Storage

As technology improvements keep increasing with regards to virtual environments there becomes a greater demand for storage area networks, or shared storage. These storage networks enable a great deal of functionality but add a layer of complexity to the entire virtual network. Traditionally there was a separate staff for storage in charge of just that. After virtualization became so important in many corporations, having a separate team just for storage became impractical. That being said, storage and system admin positions have merged together to generate a virtualization environment admin. These admins are in charge of the virtualization servers and the storage dealing directly with the virtualization hosts and nothing else. The admins sometimes can handle the individual guests but with regards to vm sprawl, it is a larger issue if there is a large user-base. The more people with access to the system there is a greater chance to have a serious issue with VM sprawl.

There are two major facets that come with virtual environments related to storage. These facets combine the performance of the array with the overall storage usage of the array. Both of these metrics are used to determine if/when more storage needs to be purchased. Currently some storage area networks have a type of monitoring solution that is shipped with the product or as an add-on. A manufacturer analysis mechanism is not necessarily a poor choice, but the integration between that and the virtualization servers and the virtual machines themselves is not possible. This combination is where the power of monitoring and visualization can be realized.

So there are multiple factors when it comes to monitoring the storage of a virtual environment. The main pieces that makeup storage in a virtual environment are over storage utilization, virtual machine disk size, and used space in the VM. By monitoring each of these variables as well as virtual machine and the virtualization servers, the admin can get a greater understanding of the environment and the interactions between all of its components.

2.4 Visualization

Data collected about virtual environments, or any system based environment is very important. The ways to go about collecting the data are all very similar. Usually a software development toolkit (SDK) or scripted command line utility is used to collect the data and store it either to a file, database, or directly to a filesystem hierarchy. Visualization of that data was very ad-hoc, reports were generate when needed. Initially this solution was suitable for the requirements.

As systems became more complex and and the data collection became more efficiency it was possible to collect more information about the environment. Not only was it possible to get more data from each component, it was also more efficient and faster as well. The requirement to collect data, and retrieve it on the fly for report generation was increasing rapidly.

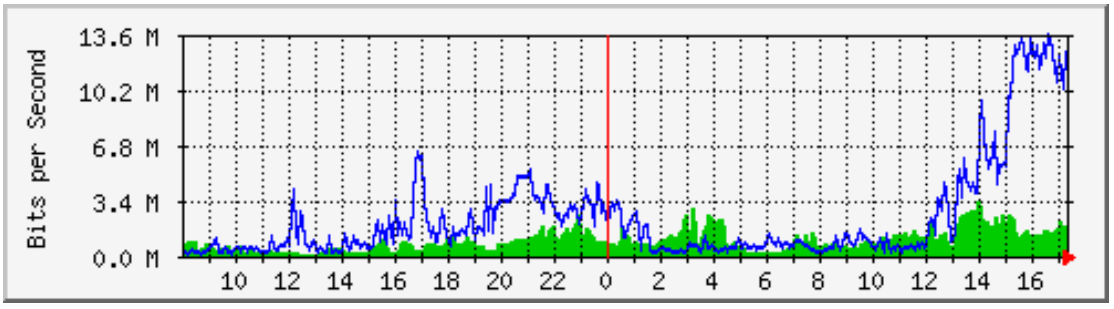


FIGURE 2.1: MRTG Graphing Output[1]

Visualization of data within systems and network environments has been very impressive over the years. The need for visualization is an obvious one and the development into applications and application programming interfaces (APIs) are a great example of this need. The rapid progression of Tobias Oetiker's graphing and data storage methods are testament to this statement. Tobias developed a product called *Multi Router Traffic Grapher* (MRTG) and it was a simple but effective mix of data collection and visualization in the same product [1]. It queried routers over the simple network management protocol (SNMP) and graphed the output. A snapshot of the output is in Figure 2.1.

After a while it was apparent that MRTG did not have enough flexibility to complete other visualization tasks, which it was not designed for. Tobias developed an application called *RRDtool*, RRD stands for round robin database. RRDtool was not actually used for visualization but a rock solid way to store data, visualization code was opened to

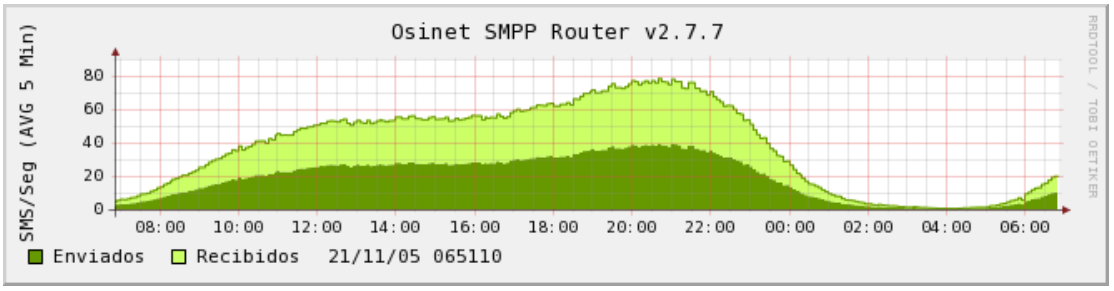


FIGURE 2.2: RRDtool Graphing Output[2]

the administrator and not provided by the developer for maximum flexibility [2]. A screenshot of what can be done with RRDtool is shown in Figure 2.2.

While RRDtool had increased flexibility when compared to MRTG it was still limited for the use in this thesis. In order to support a web application and analysis in JMP a relational data was required. Storing the data in a relational database allowed for quick access to very large amounts of data.

2.5 Conclusion

When combining data collection from multiple sources and visualization visualization a unique challenge crops up. It is important to determine the most appropriate tools for this job. In this case MySQL[4] was used because of its speed and efficient handling of large data sets. The graphing methods would need to be extremely flexible considering the different data types and varying amounts of data passed to it. Using amCharts [5] proved to solve all of those problems because it understands XML and can create interactive graphs of almost any data passed to it.

3

Methodologies

3.1 Overview

Multiple visualization techniques were implemented to improve the usefulness of the data collected. In order to convey the importance of understanding each component of a virtual environment as well as their interactions between each other a platform that could be simple yet descriptive was used. Improving the visualization of a virtual environment will increase the manageability, which should in turn increase the ability to reduce the number of virtual machines and/or improve performance of the environment as a whole.

3.2 Environment

In order to develop a procedure to analyze and maintain a virtual environment, a complex environment was used for this thesis. All of the data collected for this thesis was polled from a production network in use 24/7. There are about 500 virtual machines and growing within this virtual environment.

While every virtual environment is different, there are similar aspects within each and every one. Every environment can be broken down to three major parts; storage, servers, and network/fabric. The environment consisted of multiple servers interconnected with two different types of networks. One network used solely for the shared storage and

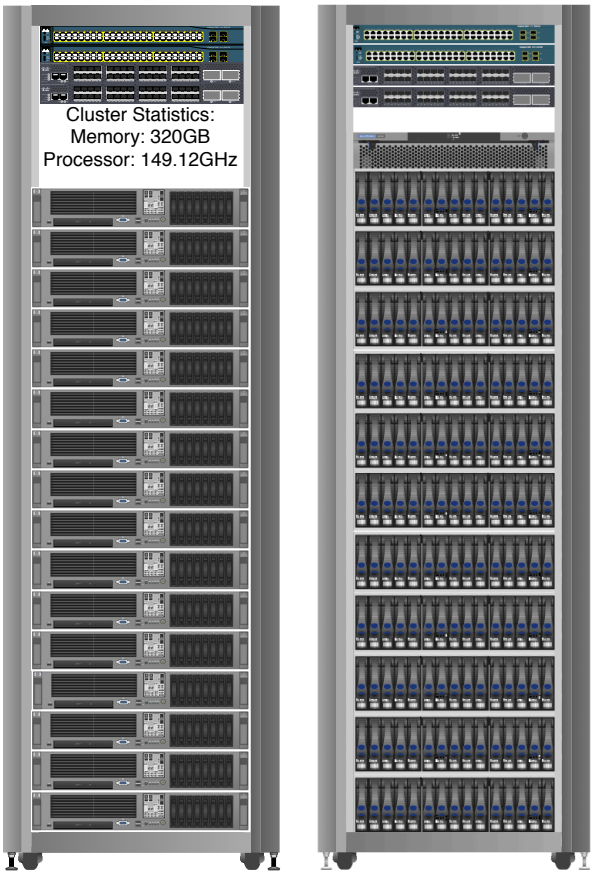
another network used TCP/IP traffic. The table 3.1 lists the hardware used in the environment.

Component	Classification	Quantity
HP DL 380 G5	Server	43
EMC CLARiiON CX4-240	Storage	1
Cisco Catalyst 3750	Network	4
Cisco MDS 9134	Fabric	4

TABLE 3.1: Virtual Environment Components

A full VMware cluster is made up of 16 servers in the environment. There are redundant switches at the top of each rack, two Cisco MDS fiber channel switches and two Cisco 3750 multi-layer switches. The table 3.1 defines a a fourth set of switches which are used to the EMC array connection into the environment. The layout of a rack used in this virtual environment cluster is shown in Figure 3.1

FIGURE 3.1: VMware Cluster Rack Layout



(a) Cluster Rack (b) Storage Rack

FIGURE 3.2: Rack Layouts

To dive deeper into the environment used we can combine what we know about the servers used within the environment and compile a maximum resourcelevel with regards to the environment as a whole. Shown in the table 3.2, we now have an environment resource total.

Hardware	**Value**	**Environment Total**
Servers	43	43
Processors/Server	2	86
Cores/Processor	2	172
Core Clock Speed	2.33 GHz	400.76 GHz
Memory/Server	20 GB	860 GB

TABLE 3.2: Virtual Environment Hardware Specifications

By creating compiling these numbers we now have a ceiling to compare the results to. Due to the nature of virtual environments and their inherent flexibility with back-end resources, it is certain that it will change. When the back-end of the environment changes, either add more servers or replace old servers, the resource ceiling will change.

3.3 Purpose

The purpose of this thesis was to provide a set of tools that will allow for the enhancement of the management of virtual environments. The enhancements are in the form of improved monitoring via statistical analysis between each of the components that makeup a virtual environment. To further the impact of the data collected during the monitoring process of a virtual environment, different visualization techniques were used.

3.4 Limitations

There will inevitably be limitations in this thesis project due to the scope of the project. Certain limitations will be more prominent only due to the factor of time. Some of these limitations that may arise will be regarding speed, encrypted communication, and number of potential visualization platforms.

Considering the most important process in this project is regarding data collection, analysis, and visual representation of a virtual environment, that will be the highest priority. The data collection itself can yield a great deal of information, if there is too much data initially, a pruning method or random sampling process will need to be implemented.

Once data collection and analysis is complete some of these initial limitations, if they exist, will be addressed. For example, with regards to number of visualization platforms, after one platform is functional, then and only then will a second platform be explored and implemented, time permitting.

There will be many different processes that are dependents of analysis of the virtual environment. That being said, if one or more of these processes experience any latency, it will slow down the entire analysis of the process. If the analysis takes longer than usual, optimizations will be implemented as needed and time permitting.

3.4.1 Assumptions

There were some assumptions that had to be made during the course of this thesis. These assumptions were necessary in order to ensure the accurate interpretation of the data collected. It was assumed that the storage array being polled was an EMC[2] CLARiiON storage array using the Navisphere[1] manager. Secondly it was assumed that VMware[2] was ESX[3] was used on all the virtual environment servers.

3.4.2 Validity

The data collected and analyzed for this thesis is strictly applicable only to this environment at the time of the data collection. Due to the fast growth of the environment outlined in section 3.4.3 it would be unlikely for data collected in the past to apply to a current environment. That being said using past data would prove useful when compared to current results to detect trends or improvements in the environment.

[1] EMC[2] Software Management
[2] Virtualization Technology Leader
[3] VMware Virtualization Operating System

3.4.3 Growth

Growth can be defined in two ways with regards to a virtual environment, back-end growth and front-end growth. Back-end growth would occur when more resources were required after analysis was completed and determined that no virtual machines could be consolidated. Other reasons why back-end growth could occur is when servers fail, and new servers which are more powerful replace the older servers.

Front-end growth on the other hand is referring to virtual machines, as number of virtual machines increase, storage, memory, and cpu resources are consumed. When virtual machines are created without bound, these resources can be consumed at an astonishing rate.

The ideal option would not be to prevent growth within a environment but control and manage the changes and additions to the environment. Processes can be put in place to track the growth of the environment. With this in place it would be easier to manage the changes within an environment and to catch problems before they occur.

3.5 Procedure

By using a custom architected web application along with a statistical approach leveraging detailed visualization one should be able to improve the performance of virtual machines and efficiency of the storage within a virtual environment. The improvements will be measured by disk usage, CPU usage and memory usage.

Data collection will run continuously pulling from ESX servers as well as the storage system in a virtual environment. The data collected will be stored in a relational database that can be leveraged by many applications. By utilizing statistical approaches outlined in JMP Start Statistics[6] one should be able to improve the virtual environment. Once the data is collected, the analysis should be able to identify any potential trends and interactions. Once the analysis of the data is complete the visualization of the results will be initiated.

3.5.1 Data Collection

Both the servers and storage have their own proprietary collection methods, each with access restrictions and separate application programming interface (API). Each of the devices collect data at similar intervals, the servers get polled every 5 minutes and the storage gets polled on an average of every 3 minutes. The data collected within the devices gets stored during the data in a persistent data store and will be polled at the end of every day through the API.

Each component required the use of a different API[2] leveraged from a separate script. Each script or set of scripts depending on the component of the environment added the data collected for that day into a relational database. The database used for this thesis was MySQL[3]

The VMware API has a great deal of features regarding performance data collection. Details can be collection ranging from IOPS on a server to IOPS on a VM level. The level of detail is an advantage as well as a disadvantage. In order to work through the complexity of the API there are some techniques that must be used. For example when collection data from a continually polled instance like a vCenter server it is import to collect only the data necessary and using all the data collected [7]. If the data that is collected is not used then the time taken to collect the data was lost and thus decreases the perceived performance of the web application, not to mention the amount of storage used will increase.

3.5.2 Visualization

Disregarding the topic of virtualization for the time being, it has always been the goal of network and system administrators alike to collect data about their systems. The data collected allowed the admin to get a better idea about how their systems were operating. Having past data to compare current operating conditions against aids in the detection of potential problems and other degradations in systems. With regards to virtualization, effective data visualization is extremely important. There would be no effective way to monitor a virtual environment without data collection about the virtual machines, the virtualization hosts, and the storage system.

[2] Application Programming Interface
[3] Well-supported and actively developed relational-database.

Typically a virtual environment will be polled during a normal duty cycle and not much experimentation is performed ahead of time. That being said one of the most effective methods to visualize the data in a production environment is to use a *time series* representation [6]. The reason this method works so well is because usually when in production, there is little room for experimentation. The time series representation will allow one to potentially forecast future trends to have a better idea of growth regarding the virtual environment.

Visualization should not stop at time series representations of data, especially with virtual environments. There is more information that can be determined from the data that is collected. Some of the other visualization techniques that can be used are bubble plots, control charts, and overall distributions of the data. All of these methods have their own advantages and can all be used to gain a greater understanding of the environment as a whole. One visualization technique to find all the possible corrections in the virtual environment called the *scatterplot matrix* in the JMP software would be beneficial for the analysis[8]. By finding correlations between the data sources within a component we can then optimize our analysis by knowing more information about the environment.

3.6 Conclusion

By leveraging effective visualization of data using both a web application and the JMP product, there is much more value that can be drawn from the raw data. By using some of the JMP platforms outlined in [9] there are many meaningful types of output we can provide about a virtual environment. The output that JMP will provide will offer a more detailed and customizable viewcompared to the web application. Aside from being more flexible and agile, JMP requires an additional piece of software with its own learning curve. That being said the web application offers a great way to have the virtual environment visualizations available to a wider range of stakeholders.

4

Web Application

4.1 Overview

As stated in section 3.1 there are two major parts to this thesis, data collection and data visualization. The data collection was handled by separate scripts for each component of the virtual environment and the data visualization was handled by a custom web application and JMP. The following section describes the custom web application written for this thesis.

A web application for ongoing analysis allows for a platform independent approach for rapid analysis of a virtual environment. Not only is this an efficient approach to provide admins with the most up-to-date data but it is also a fantastic way to deliver the results to a wide range of users as efficiently as possible.

Knowledge of virtualization has become mandatory within the field of system administration. It is important to not only posses the knowledge of building a virtual environment but the ongoing maintenance and management of one. By providing a set of tools and procedures it is then possible to manage a very large environment with ease compared to a manual style of management.

Monitoring a virtual environment is a very important task in order to maintain optimal performance. In order to effectively and accurately monitor a virtual environment it must be a quick operation to initially see the health of the environment. If further

investigation is required then a more in-depth process can be documented and easily performed.

Providing a web application to offer a way to monitor the virtual environment gives the administrators insight into the environment. By simply having the web application available to the administrators, they have the ability to quickly discover where issues exist within the environment without time consuming analysis. This allows the administrators to start debugging quicker. If the web application is used on a regular basis it would even be possible to catch problems before they start and pro-actively perform maintenance on the environment.

The web application is named *veViz* which stands for **v**irtual **e**nvironment **Viz**ualization. The web application itself is a front-end to the data collection methods. These methods are supported by a few backend scripts which get called from a cron[1] job. By using a front-end which reads directly from the database which is filled via different backend scripts, it provides the ability to use the native API of each component of the virtual environment. Using the native API where available is the best choice when collecting data from a virtual environment because it enables the admin to collect the data with the most detail possible. This was a much better approach than the initial option of using the `esxtop` command. The VMware `esxtop` command was resource intensive and required communication to every ESX server. Using the API instead allows a central collection point from the vCenter server itself.

4.1.1 Database

The database schema was designed to support both a web application as well as more in-depth analysis via JMP. That being said, the amount of normalization was minimal due to the complexity it would add to the third party analysis. There were four tables used for the development of the application, two for summarizations of the data and two for raw data collected.

Summary Data	**Raw Data**
array_summaries	storage_arrays
host_summaries	hosts

[1] Cron is a daemon in *NIX which runs scheduled tasks.

The summary tables are used for the dashboard of the application and are appended to via a script within the web application detailed in Appendix D. The other two tables are master tables that receive data directly from their respective sources. The VMware ESX API fills the hosts table via a perl script outlined in Appendix E and the storage_arrays table gets filled via a CSV import using the EMC^2 Navisphere Analyzer CLI via a script detailed in Appendix H.

Refer to Appendix F.1 for the detailed database schema design.

4.1.2 Interface

The web application itself is modular considering the fact that it brings together each piece of a virtual environment to one concise monitoring interface. The interface was designed to be clean and simple without many options to limit confusion. The three major pieces of the web application are the dashboard, hosts view, and storage array view. Each of the pieces of the web application enables their own specific purpose to allow for better visualization of that specific component in the virtual environment.

4.1.3 Dashboard

The application has a home page view that quickly will summarize the most important factors of the virtual environment. The dashboad of the web application allows for a summary the virtual environment based on a pre-calculated average of each value collected that that component. The dashboard page updates each day when the cron script is enabled. A full page overview of the dashboard is shown below in Figure 4.1.

The dashboard consists of two major aspects of the virtual environment, the storage and the servers. It summarizes both of these in an individual graph for each section of the environment averaged out for each day. For example, the server side of the environment is made up of 43 servers, these servers are polled every 5 minutes. The dashboard takes every value for each server and averages them for that day. This simple front page gives a very brief overview of the operational characteristics of the environment as a whole.

4.1.3.1 Available Counters

There are a set of counters that were used most prominently to determine the operational characteristics of the environment. The counters chosen for this task were different for

FIGURE 4.1: veViz Dashboard

the storage array and ESX servers and are represented in Table 4.1. Having different counters was done mainly because of API restrictions of the respective environments.

Array Counters

Counter	Unit
Utilization	Percent
Queue Length	Integer
Response Time	ms

Host Counters

Counter Name	Unit
CPU Usage	MHz
Memory Usage	Percent
Disk Usage	KB/sec

TABLE 4.1: Available Environment Counters

Each environment consisted of 3 main counters which summarize the operation of the specific environment to a degree. The three counters used for each environment best

represent the performance of that environment well. A more detailed view of the environment can be gained by reviewing all the counters available for that environment. This would not be possible within the web application due to performance reasons.

4.1.3.2 Host Summary

The host summary polls the VMware vCenter database for the counters in Table 4.1 which are collected nightly. These three counters are the best counters available at the ESX host level to determine the performance of each ESX host within the environment.

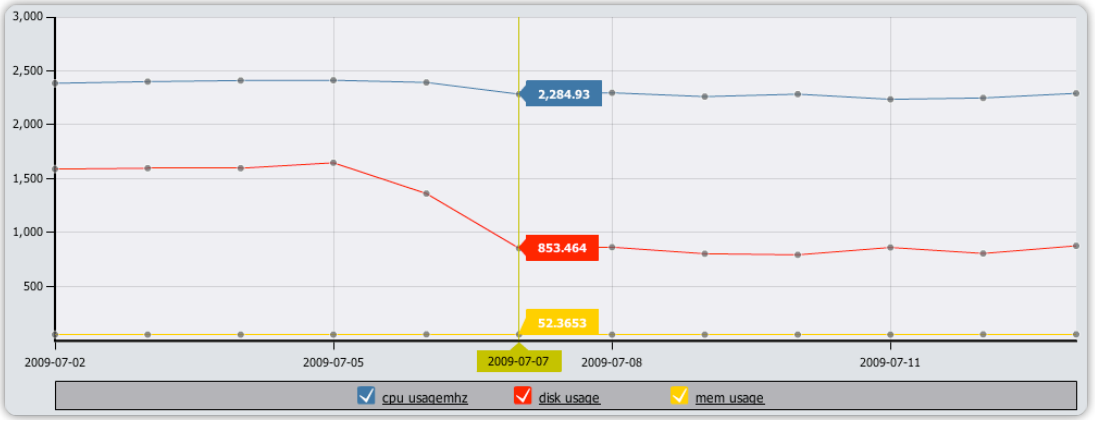


FIGURE 4.2: veViz Host Summary Detail

4.1.3.3 Array Summary

The counters used for the dashboard are pulled from the database using each service processor (SP) as the object to query. The service processor handles all the transactions on a storage array so it is a good place to start when querying the usage of a storage environment. That being said, there are many other objects in a storage array where performance contention issues may occur. Analysis of these objects are available in the storage tab of the web application described in Section 4.1.6. Table 4.1 shows the three main counters used in the dashboard view for the overview analysis.

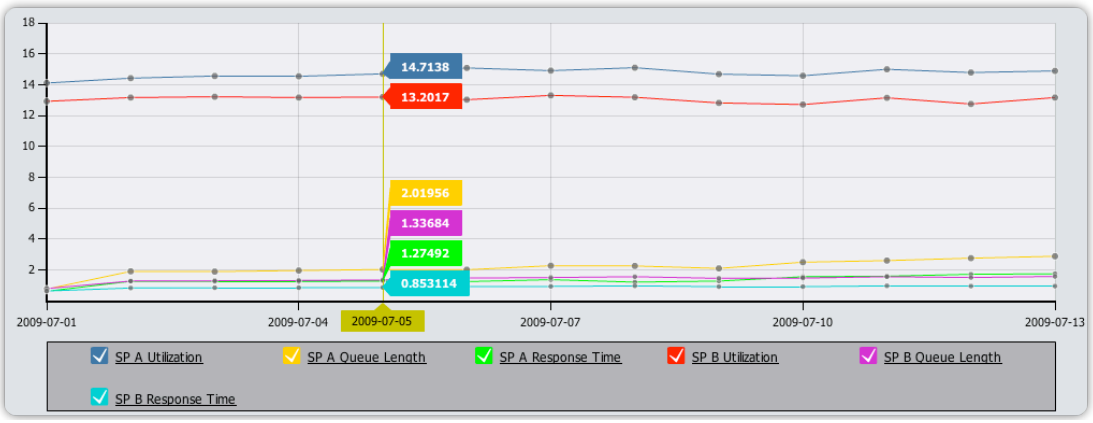


FIGURE 4.3: veViz Array Summary Detail

4.1.4 Navigation

Navigation is a very important aspect of any application. Without a simple navigation tree it would be difficult for a wide audience to use any application. Considering that this application should be able to be used by anyone that is interesting in the virtual environment, there should be a rock solid navigation implemented. Below in Figure 4.4 shows the navigation used in the web application.

FIGURE 4.4: veViz Navigation Menu

Along with the dashboard for the web application there are detailed views for both hosts and arrays. By clicking on one of those links you will be brought to the selection screen.

The virtual machine tab is disabled for the time being, until data rollup is implemented it would not be practical to implement collection performance data for each virtual machine in the virtual environment.

4.1.5 Hosts View

A detailed view of each host is possible within the web interface. By providing the option to drill down to each individual host allows for a better understanding of each host. Not only can an individual hosts details be viewed but the user can select any available date range to see anything from daily trends and beyond. These options are selected in a fashion as depicted in Figure 4.5.

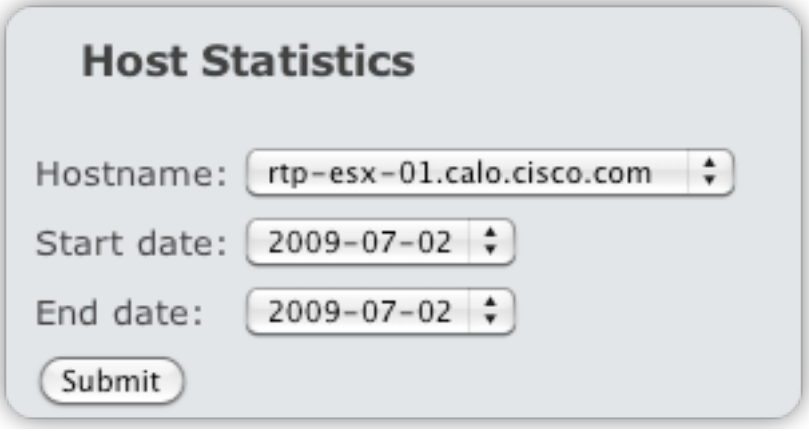


FIGURE 4.5: veViz Host Selection

Once the users selects the host and time period which is of interest after submitting the values to the application it will return the desired query. For every host that is selected three counters are displayed by default. These counters can be hidden to see a better representation of a specific counter. Figure 4.6 is the display of all values on one chart.

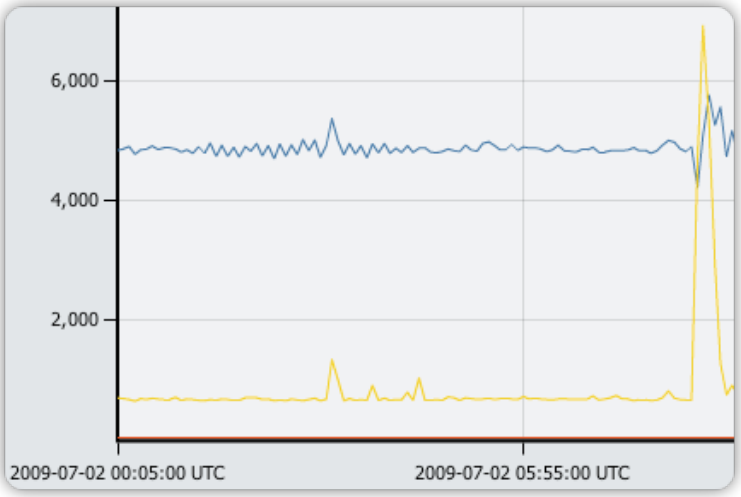


FIGURE 4.6: veViz Host Detail

The units are displayed below the chart in a legend. See Figure 4.7 below.

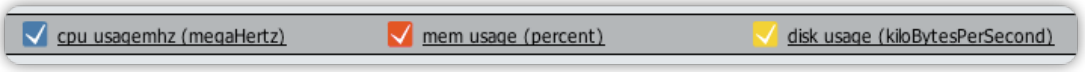


FIGURE 4.7: veViz Host Legend

If you want to only view memory usage uncheck the boxes in the legend for the chart. By doing this the chart will only the data that is under the series that is checked. The chart will redraw automatically using the new data set without having to reload the entire page. Below in Figure 4.8 shows the level of detail that is gained for the memory counter by limiting it as the only data on the graph.

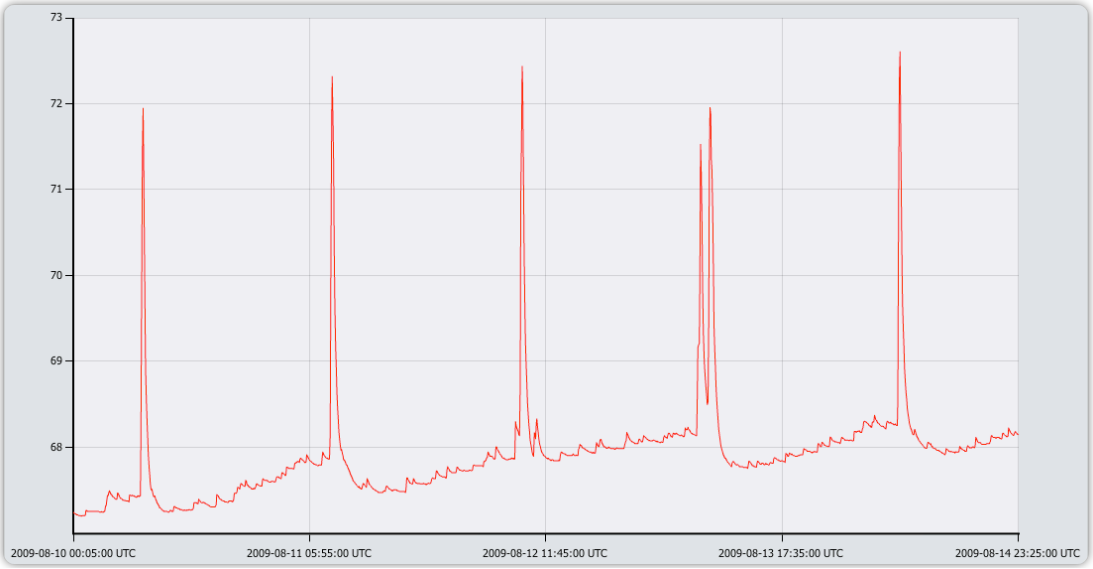


FIGURE 4.8: veViz Host Memory Usage

4.1.6 Storage Array View

The storage array used in this virtual environment was an *EMC*^{2} *CLARiiON CX4-240*, there is a command line interface which allows for secure communication between a EMC^2 provided application and the array. On top of that the collection daemon can be set to run indefinitely or for a set amount of days. Once we download the data for each day there is some post processing that needs to be done before we can bring it into the web application's database. We must sanitize the date format to be compatible with the MySQL default format, on top of that we need to remove the first line of the file because those are the column names. Once this is done we are ready to import the data into the database.

The web application itself reads the contents of the database just like the Hosts selection screen, this allows for the form to be dynamically generated based on the content of the database. By dynamically generating the form fields the web application could then support any number of storage arrays just be changing the back end scripts to collect the data along with the database schema. A view of the selection screen is shown below in Figure 4.9.

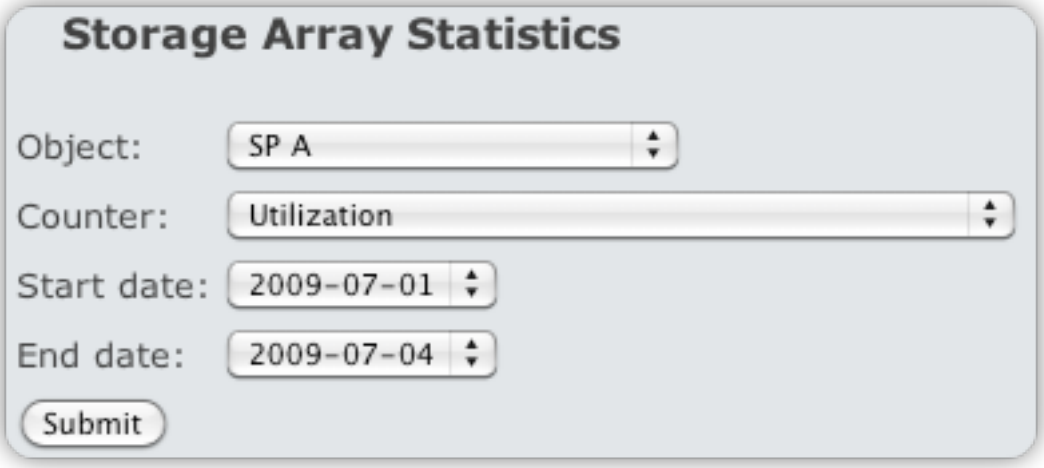


FIGURE 4.9: veViz Array Selection

Once major difference with the storage array and the hosts selection screen is that you now have the ability to view a certain counter. By doing this there was little post processing that needed to be done on the storage arrays data. Once the data is in the database we can view each counter individually. Below in Figure 4.10 shows the SP A Utilization.

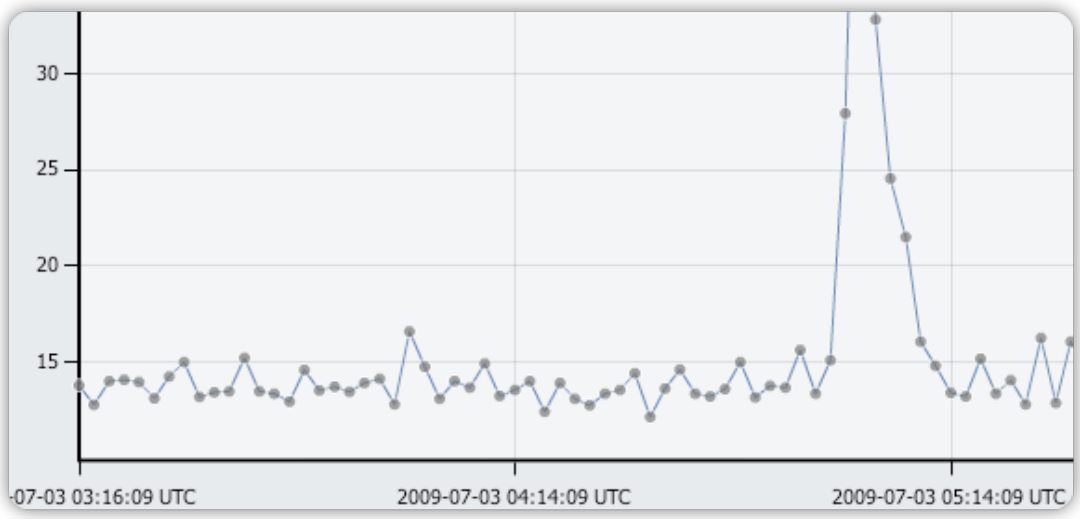


FIGURE 4.10: veViz Array Detail

In contrast to the host detailed view, with the storage arrays view only one counter is available for viewing due to the increase amount of data that is collected. It is not feasible to have a great deal of data within these charts because the readability and speed of the chart generation decreases dramatically.

Similar to the hosts view, the legend displays the counter that is being represented in the graph above and the ability to hide the line.

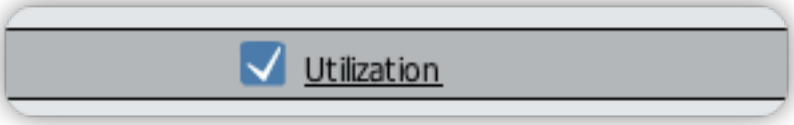


FIGURE 4.11: veViz Array Legend

4.2 Analysis

The host portion of the web application accomplishes two tasks, providing a daily summary of the virtual environment with regards to the three counters being polled as well as a detailed analysis of each host being monitored. By offering the two levels of details trouble spots can be found using the daily summary and then investigated in great detail on a host-to-host basis.

In order to keep the web application performance and resource utilization at a reasonable rate this was decrease to the main players in the analysis. The storage monitor in the

web application has two parts as stated in 4.1.3 and 4.1.6. Each of these sections play an important role in the ongoing analysis of the storage component.

While the web application is a fantastic way to dynamically generate charts on the most current data, there is greater detail that can be learned if a move in-depth analysis is done. The application used for the analysis is JMP[2]

[2]JMP is a division of SAS. `http://www.jmp.com/`

5

Results & Discussion

5.1 Overview

The two pieces of the environment which were monitored were the ESX servers and the EMC storage array. Each of these parts of the virtual environment are integral pieces that must be maintained. The host statistics pulled via the API consists of the three major counters. The counters that were polled via the API from the virtual environment are listed in Table 4.1. In order to improve the visualization of the data post-collection modifications have been made within JMP.

A detailed view into the environment along with the web application allows for the ability to drill-down into the environment on trouble spots. By having this magnifying glass on top of using the web application an administrator can potentially track down problems which have not manifested by failure and remedy them without the end-usereven knowing there was a problem in the first place.

5.2 Host Analysis

Knowing a virtual environment is extremely important, especially when monitoring the performance and capabilities of the environment. There should be an inventory of components within the virtual environment consisting of; number of servers, processors in each server, cores per processor, and memory per server. Having this will provide a

baseline of the environment that can be tracked as well as used to predict capacity requirements when more servers are needed. All that being said, refer to Table 3.2 for the information on the virtual environment used for this thesis. The last column in Table 3.2 shows the summation of all the available resources within the environment, we can use these numbers to compare our findings to. The data sources that were polled from each host were CPU usage (Section 5.2.1), memory usage (Section 5.2.2), and disk usage (Section 5.2.3).

5.2.1 CPU Usage

CPU usage within a virtual environment is extremely important. CPU resources are important to have plenty extra of because virtual machines can have a spike in CPU usage at any time. The quartiles shown in Table 5.1 show that while normal operation is around 1 CPU in total MHz, spikes can jump to a much higher extreme. While trends can be calculated to better estimate when these spikes will happen, there should always be plenty of available CPU resources.

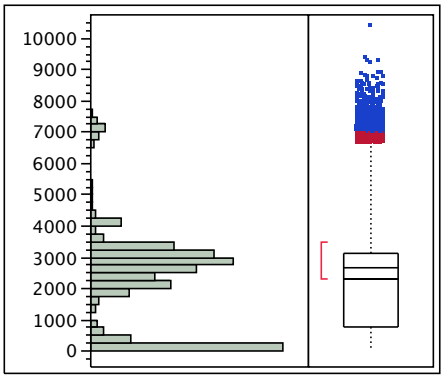


FIGURE 5.1: CPU Usage (MHz)

Statistic	**CPU (MHz)**
Mean	2327.0665
Std Dev	1544.874
Std Err Mean	3.9164614
Upper 95% Mean	2334.7427
Lower 95% Mean	2319.3903
N	155596

Quantile	**CPU (MHz)**
100% (maximum)	10373
75% (quartile)	3136
50% (median)	2661
25% (quartile)	774
0% (minimum)	0

TABLE 5.1: Host CPU Counter Moments

The CPU counter has been further split up into 4 groups or levels. These levels provide more insight into the distributions in the figures below. The CPU levels are designated by the color of the data points. In order to color the fields in all of the charts a data filter was used in JMP to first make the selections. The ranges that were used are outlined in Table 5.2, once the data filter was applied a categorial value was pasted into the selected rows. Each value of the column then had a color associated to it.

CPU usage related to this virtual environment is relatively in control, two ways to quickly visualize this is to categorize each and color each CPU level which was done according to Table 5.2 and then run a distribution of hostnames vs. CPU levels by hostname as shown in Figure 5.2.

Color	CPU Level
	0 MHz - 2333 MHz
	2334 MHz - 4666 MHz
	4667 MHz - 7000 MHz
	$\geq$ 7001 MHZ

TABLE 5.2: Host CPU Level Color Legend

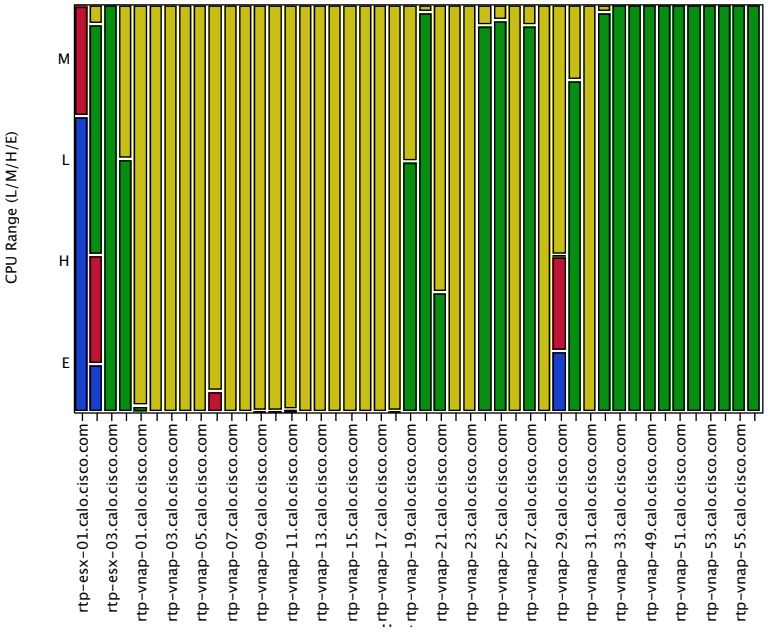


FIGURE 5.2: Host CPU Usage by Host

As you can see from the figure that the environment is very well balanced with regards to CPU usage. The hosts with a high number which were added to the environment at a later date than the lower numbered hosts have an obviously lower CPU usage. Figure 5.3

shows another way to look at the distribution of CPU usage is to look at the percentage that falls within each level.

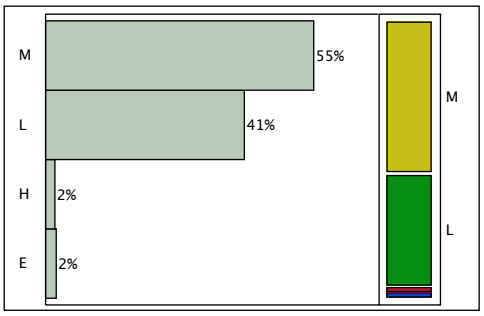
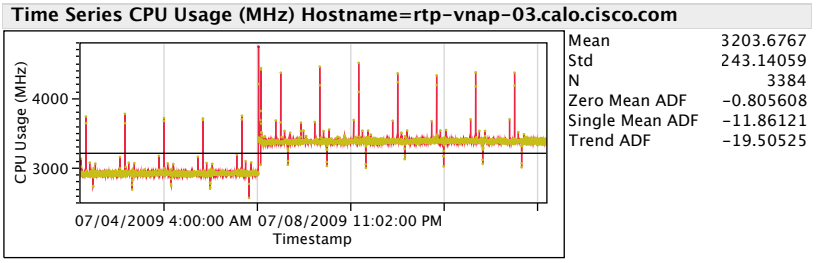
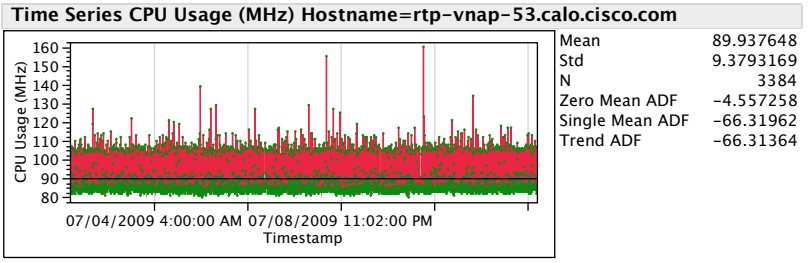


FIGURE 5.3: Host CPU Usage Range

When looking at the statistics collected for the entire environment a few assumptions can be made about the mean operational levels of the servers within the environment. As shown in Table 5.1 it can be stated that on average the mean CPU usage in MHz recorded is about the frequency of one CPU core per server in the environment. By investigating server, not all servers are running at a similar level as shown in Figures 5.4(a) and 5.4(b).



(a) rtp-vnap-03



(b) rtp-vnap-53

FIGURE 5.4: Host CPU Time Series

Each of the servers above show a large difference in CPU usage this is due to the level of activity of each VM running on the servers as well as the number of VMs actually running on the server. Maybe setting a warning and critical threshold for each server could aid in notifications of over-utilization of a single server and allow for load balancing.

5.2.2 Memory Usage

Memory is the most important resource regarding the servers in a virtual environment. There are technologies that VMware ESX leverages to conserve RAM within like VMs. That being said, it is not predictable nor is it a replacement to physical memory. Looking at Table 5.3 we can see that the mean memory usage, an average of 52 percent of the RAM on each server is being consumed in this sample. Figure 5.5 does a great job illustrating the memory usage within the virtual environment.

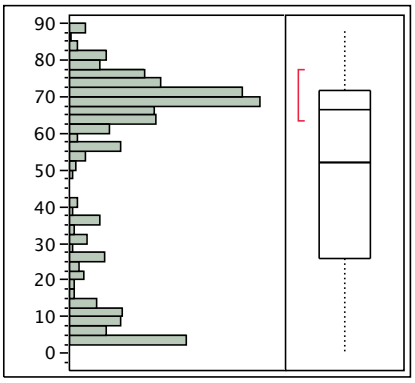


FIGURE 5.5: Memory Usage (%)

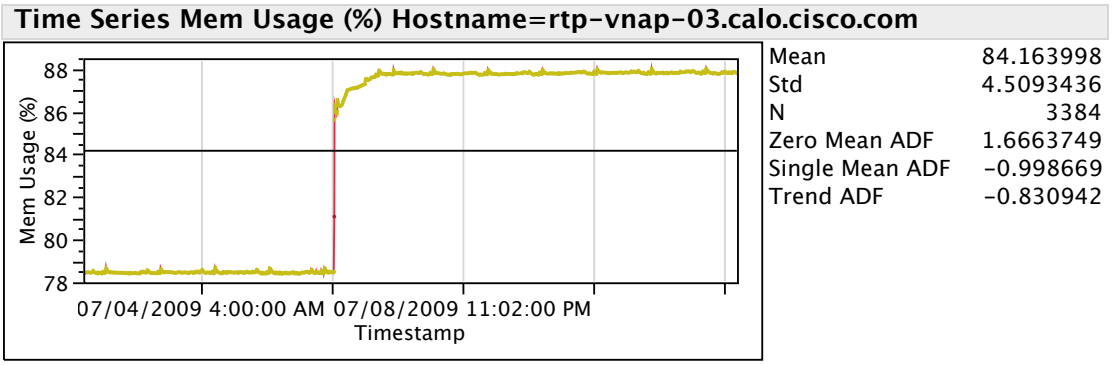
Statistic	**Memory (%)**
Mean	52.062792
Std Dev	27.148344
Std Err Mean	0.0688247
Upper 95% Mean	52.197687
Lower 95% Mean	51.927897
N	155596

Quantile	**Memory (%)**
100% (maximum)	88.03
75% (quartile)	71.54
50% (median)	66.41
25% (quartile)	26.04
0% (minimum)	0

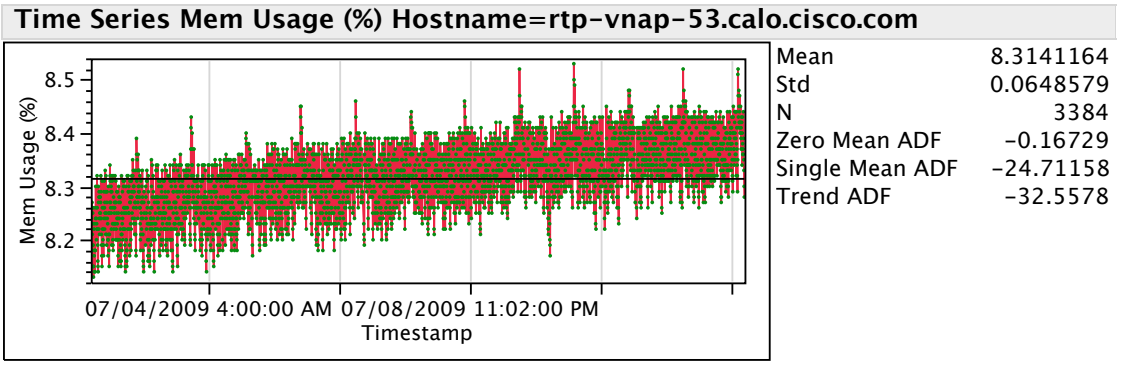
TABLE 5.3: Host Memory Counter Moments

Memory usage for this virtual environment while being stable is bi-modal. The distribution as shown in Figure 5.5 represents a bi-modal data set. While the memory usage for

the environment is relatively stable there are two things to mention here. Most importantly there are some hosts which are heavily loaded and some hosts which have almost no memory usage at all. Secondly if the hosts which have little or no memory usage were to be removed from the dataset we would get a more accurate representation of the data. Another way to view the data is memory usage for each host, this will provide detailed information over time for each individual host. Figures 5.6(a) and 5.6(b) both have two different stories attached to them.



(a) rtp-vnap-03



(b) rtp-vnap-53

FIGURE 5.6: Host Memory Time Series

Take rtp-vnap-03 for example, the memory usage on this host is extremely high compared to rtp-vnap-53. The reason for this is rtp-vnap-53 was added to the environment at a much later date and does not have as many virtual machines associated with it.

To provide a good overview of the entire virtual environments' memory footprint of each host, we can perform a bivariate analysis between memory levels defined in Table 5.4 which provides a chart as show in Figure 5.7. This chart is especially useful to quickly

identify hosts that a dangerous amount of memory being consumed and allow for relation or re-assignment of these memory offending virtual machines.

Color	Memory Percentage
	0% - 24%
	25% - 49%
	50% - 74%
	75% - 100%

TABLE 5.4: Host Memory Level Color Legend

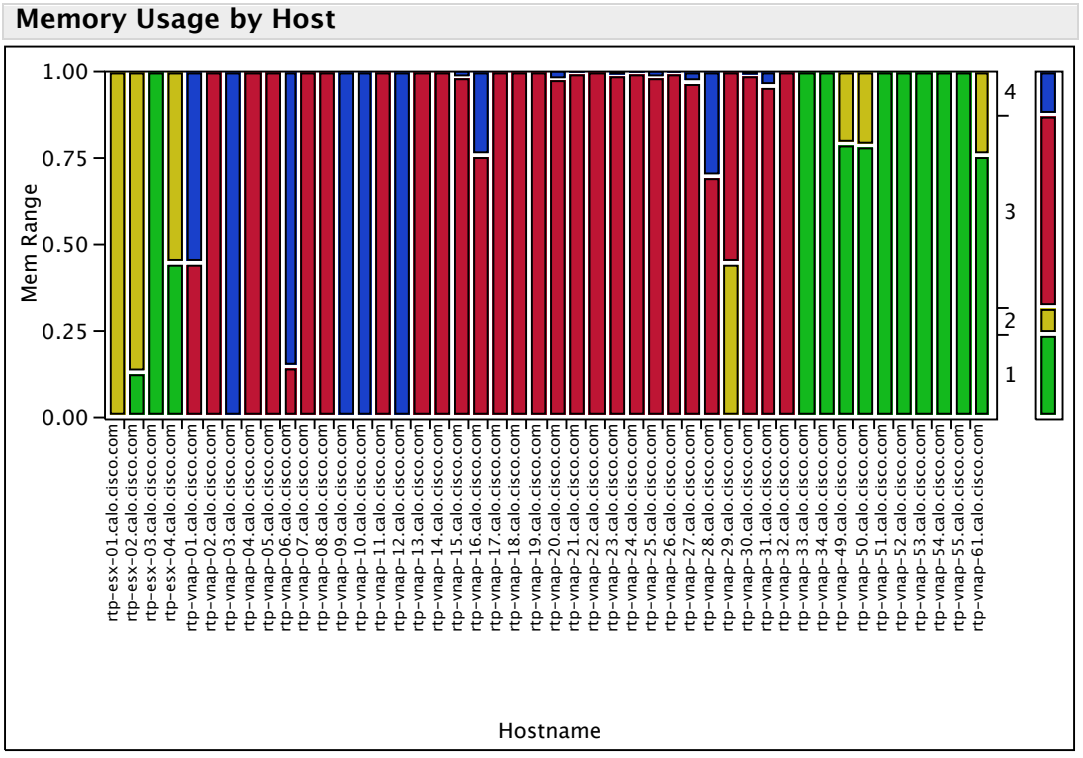


FIGURE 5.7: Overall Memory Usage

A solution to this unbalanced bi-modal server memory usage would be to provide an automated migration of the virtual machines to a lightly loaded server as opposed to the server with high memory usage. VMware provides this capability within a cluster, but not between clusters. So for this environment running an analysis like the time series charts in the figure above weekly would provide the information to the administrators necessary to load balance the virtual machines accordingly.

5.2.3 Disk Usage

Disk usage when polled from each server can be quite diverse. There are times when servers have most of their VMs shutdown, during these times the disk usage will be 0. There are also times when a large portion of the VMs running on that host are powered on and start thrashing the storage array, this can be seen when looking at the quantile statistics in Table 5.5 and the distribution of the data in Figure 5.8.

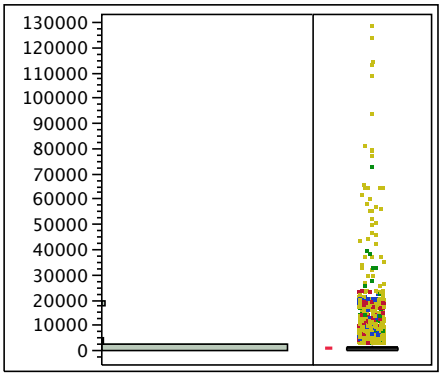


FIGURE 5.8: Disk Usage (KB/sec)

Statistic	**Disk (KB/sec)**
Mean	1137.1612
Std Dev	2652.9798
Std Err Mean	6.7256573
Upper 95% Mean	1150.3433
Lower 95% Mean	1123.979
N	155596

Quantile	**Disk (KB/sec)**
100% (maximum)	128007
75% (quartile)	1185
50% (median)	818
25% (quartile)	271
0% (minimum)	0

TABLE 5.5: Host Disk Counter Moments

While the disk usage data is an important factor within the environment, due to its arbitrary context on the servers its usefulness should be questioned. In this environment all of the virtual machines are running off the storage array so any number for disk usage on the servers is going to be duplicated on the storage array as well. There is not a correlation with disk usage and cpu usage or memory usage. The web application monitoring of disk usage should be sufficient in the analysis of disk usage.

An interesting observation about disk usage is the variability shown in Figure 5.8 and Table 5.5. With a mean of 1137 kbps and a standard deviation of 2652 kbps, it is a

very wild data source. Take for instance if a virtual machine is patching or booting up/shutting down, the disk usage will spike. The quantile statistics prove that 75% of the data falls right around the mean with a value of 1185 but has a maximum of 128007.

The lack of correlation between is also evident by looking at Figure 5.9. You can see by the disk usage levels that do not coincide with the CPU or Memory levels.

5.2.4 Counter Overview

A quick look at the counter vs. timestamp graph in Figure 5.9 will show each data source split by CPU usage levels demarcated by the levels defined in Table 5.2. This chart does a fantastic job at detailing what the other counters are polled at when CPU is a specific level. On top of that it also shows a simple time series of the three counters for all the hosts in one view.

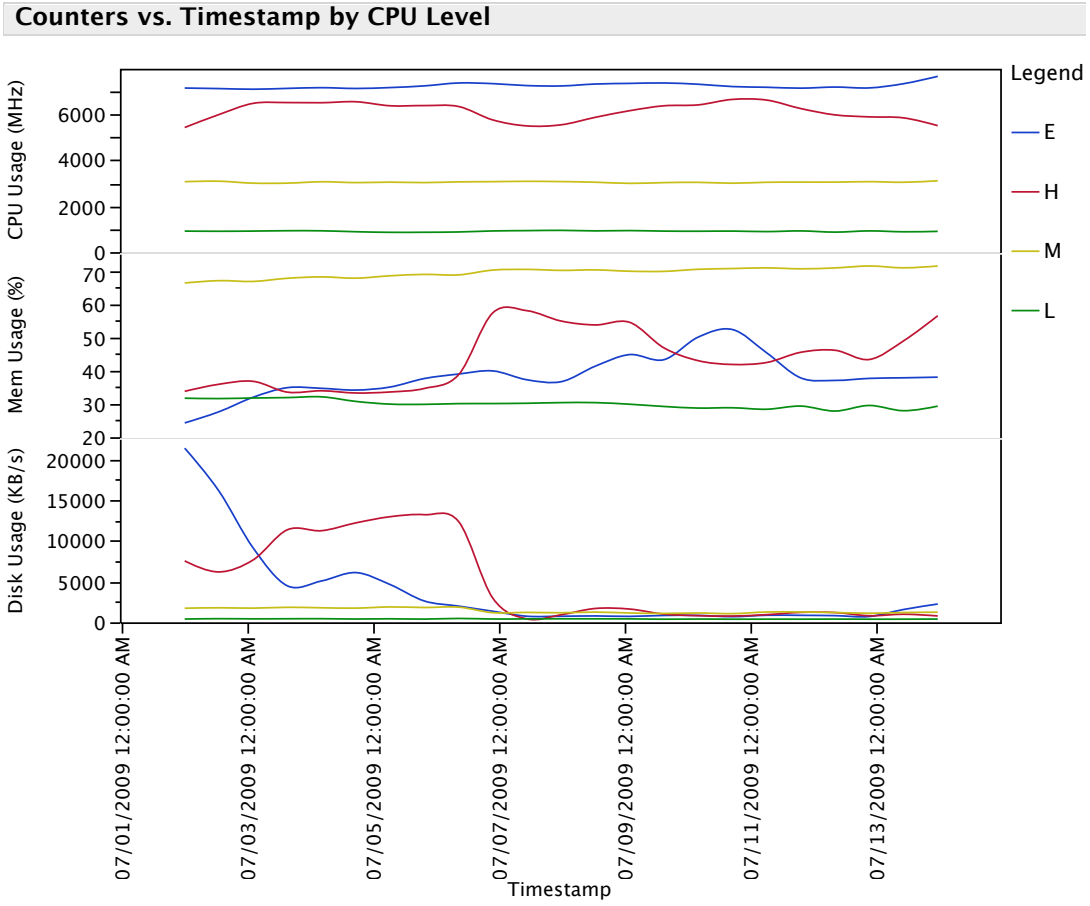


FIGURE 5.9: Counters vs. Timestamp by CPU Level

5.2.5 Host Data Source Correlations

It is always important to find relationships between any of the variables within a system. A multivariate scatterplot matrix shown in Figure 5.10 was used to determine if any of the variables collected for the servers were correlated in any way. Each of the plots show a large amount of variability in the response thus making it difficult to draw any conclusions from this chart. The r value at 0.6496 tells us there is a positive relationship between memory usage and CPU usage.

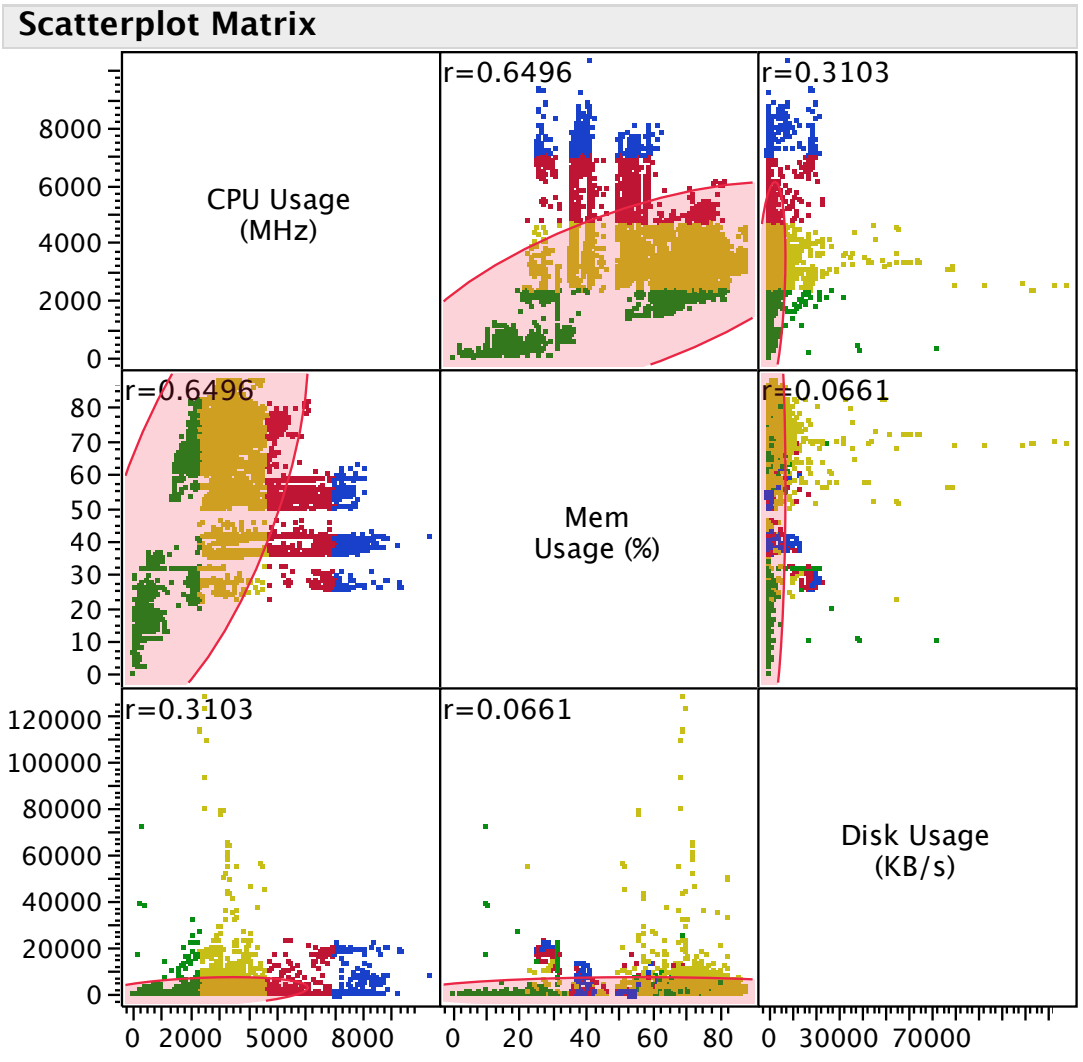


FIGURE 5.10: CPU Memory and Disk Scatterplot Matrix

5.3 Storage Analysis

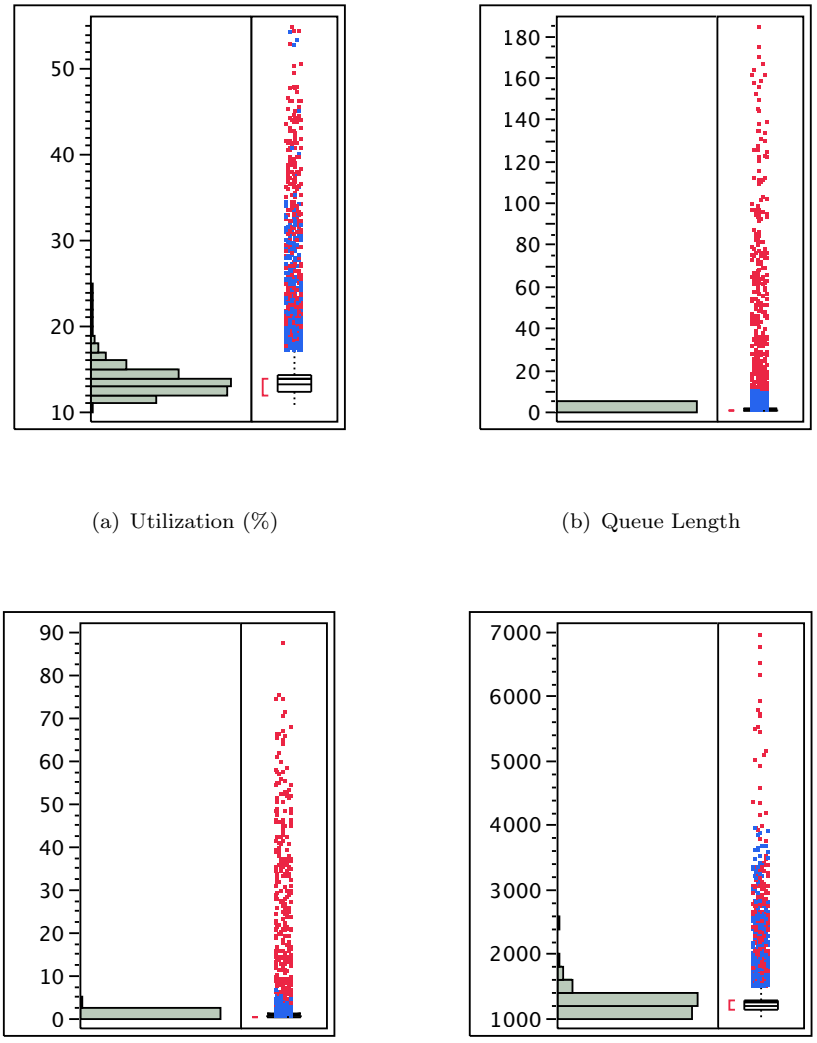
High performance shared storage is required in a virtual environment if certain virtualization technologies are implemented. For example to provide high availability and dynamic load balancing between servers within a cluster, shared storage is required. That being said, the performance of the shared storage is crucial to the environments overall health and performance. Initially the storage array had a daunting amount of variables that were polled, 123 to be exact. Most of these 123 values are collected for each object within the array. It would be impossible to monitor all 123 values for every object on a regular basis.

The web application only polled information on the service processors and used that as an overall performance metric. While this performs its task well, it is hard to know everything about the storage environment just by analyzing the service processors. There are many more types of objects that have similar data relating to the objects respectively. The objects that makeup a storage array are defined in Table 5.6. For example there could be a low load on the service processor all coming from one LUN with a LUN utilization in the 80% range or higher. The VMs running on that LUN will be performing unacceptably slow while the rest of the environment would be performing just fine. In order to determine the performance each LUN would need to be analyzed. To effectively perform the analysis on every LUN the number of counters that are polled must be reduced.

Object Name	**Quantity**
Service Processors	2
LUN	34
Disk	105
Raid Group	17
Port	8

TABLE 5.6: Storage Array Objects

The service processors of the storage array are the active/active redundant portions which control are I/O between the backend (drives) and the hosts in the virtual environment. Below are some stats collected on determinant factors of performance in a virtual environment which are outlined in the table 5.11.



(a) Utilization (%)

(b) Queue Length

(c) Response Time (ms)

(d) Throughput (IOPS)

FIGURE 5.11: Service Processor Distributions

5.3.1 Utilization

Utilization is a good metric to see an overview of the storage arrays' workload, it is not a great metric to understand actual usage. Be that as it may, it is a good way to see if there are any contention points within the service processors that must be addressed. According to the data in Table 5.7 mean utilization is at an acceptable level. Even when looking at the quartile statistics in Table 5.8 the utilization is not at a critical level.

SP A

Statistic	Utilization (%)
Mean	14.75772
Std Dev	3.0561869
Std Err Mean	0.0328035
Upper 95% Mean	14.822022
Lower 95% Mean	14.693417
N	8680

SP B

Statistic	Utilization (%)
Mean	13.0711
Std Dev	3.1295512
Std Err Mean	0.0335909
Upper 95% Mean	13.136946
Lower 95% Mean	13.005254
N	8680

TABLE 5.7: SP A/B Array Utilization Moments

SP A

Quantile	Utilization (%)
100% (maximum)	54.7093
75% (quartile)	14.876
50% (median)	14.0625
25% (quartile)	13.5225
0% (minimum)	10.8731

SP B

Quantile	Utilization (%)
100% (maximum)	53.3225
75% (quartile)	13.0363
50% (median)	12.4189
25% (quartile)	11.9736
0% (minimum)	10.3333

TABLE 5.8: SP A/B Array Utilization Quantiles

5.3.2 Queue Length

Queue length is the number of transactions that are waiting on the service processor, as the queue length increases the potential performance of the array will decrease. On this storage array the majority of the time the queue length is at a reasonable level. When looking at the quartiles in Table 5.10 it is evident that at times the queue size spikes very high. The large queue here can cause a potentially crippling performance deficit on the storage array.

SP A

Statistic	Queue Length
Mean	2.2106611
Std Dev	10.035682
Std Err Mean	0.1077177
Upper 95% Mean	2.4218133
Lower 95% Mean	1.9995088
N	8680

SP B

Statistic	Queue Length
Mean	1.4223081
Std Dev	7.5786827
Std Err Mean	0.0813456
Upper 95% Mean	1.5817647
Lower 95% Mean	1.2628515
N	8680

TABLE 5.9: SP A/B Array Queue Length Moments

SP A

Quantile	Queue Length
100% (maximum)	170.176
75% (quartile)	0.8434
50% (median)	0.7782
25% (quartile)	0.73156
0% (minimum)	0.57014

SP B

Quantile	Queue Length
100% (maximum)	183.718
75% (quartile)	0.96465
50% (median)	0.64469
25% (quartile)	0.57675
0% (minimum)	0.47223

TABLE 5.10: SP A/B Array Queue Length Quantiles

5.3.3 Response Time

Response time is a excellent metric for measuing the user-experience of your array. Making sure the response time does not get too high is imperative for the end-user. In Table 5.11 the response time is fairly close between both service processors. That being said the standard deviation is larger on service processor A. Along with standard deviation the maximum or ceiling of response time on the service processor is dangerously high as shown in Table 5.12.

SP A

Statistic	Resp. (ms)
Mean	1.36
Std Dev	5.18
Std Err Mean	0.06
Upper 95% Mean	1.47
Lower 95% Mean	1.25
N	8680

SP B

Statistic	Resp. (ms)
Mean	0.89
Std Dev	2.94
Std Err Mean	0.03
Upper 95% Mean	0.95
Lower 95% Mean	0.83
N	8680

TABLE 5.11: SP A/B Array Response Time Moments

SP A

Quantile	Resp. (ms)
100% (maximum)	75.0681
75% (quartile)	0.65509
50% (median)	0.61604
25% (quartile)	0.59116
0% (minimum)	0.25369

SP B

Quantile	Resp. (ms)
100% (maximum)	87.4407
75% (quartile)	0.78712
50% (median)	0.55387
25% (quartile)	0.51636
0% (minimum)	0.4238

TABLE 5.12: SP A/B Array Response Time Quantiles

5.3.4 Throughput

Throughput on this storage array in measured in I/O Operations Per Second (IOPS). Below in Table 5.13 and 5.14 detailed the stat summary and the quantiles for the sample data. When comparing the mean throughput on each service processor is apparent that the load is split relatively evenly.

SP A

Statistic	**IOPS**
Mean	1312.99
Std Dev	296.42
Std Err Mean	3.18
Upper 95% Mean	1319.22
Lower 95% Mean	1306.75
N	8680

SP B

Statistic	**IOPS**
Mean	1207.04
Std Dev	240.37
Std Err Mean	2.58
Upper 95% Mean	1212.10
Lower 95% Mean	1201.99
N	8680

TABLE 5.13: SP A/B Array Throughput Moments

SP A

Quantile	**IOPS**
100% (maximum)	6956.19
75% (quartile)	1312.96
50% (median)	1264.37
25% (quartile)	1203.06
0% (minimum)	1112.40

SP B

Quantile	**IOPS**
100% (maximum)	4553.52
75% (quartile)	1224.96
50% (median)	1162.06
25% (quartile)	1107.07
0% (minimum)	1023.89

TABLE 5.14: SP A/B Array Throughput Quantiles

Univariate Analysis

Up until now the storage environment has only been analyzed in a univariate fashion. While this is adequate to have an overview of each metric collected about the storage array there could be more information available to gain. Interactions and correlations about the storage array are important to uncover because that can unlock the door to faster performance profiling of the storage environment. By profiling thge storage array faster, an administrator can debug any performance related issues faster and return to an optimal state.

5.3.5 Bivariate Throughput

Utilization of the storage array, while important to know does not explain much about the environment other than the VM saturation of available array resources. We can see from the positive relationship in Figure 5.12 between throughput and utilization, the linear regression returns a $r^2 = 0.71$.

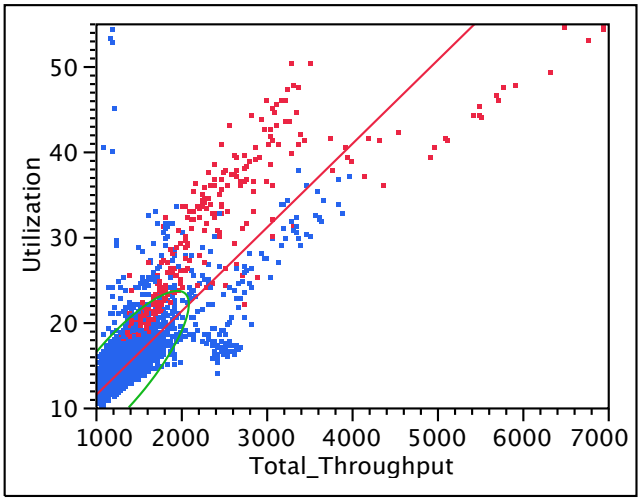


FIGURE 5.12: Bivariate: Utilization & Total Throughput

5.3.6 Bivariate Queue Length

It is clear that queue length plays a large part in the performance of a virtual environment. When there is a large amount of transactions queued up on the storage array it has a detrimental effect on the performance of all other aspects. Considering how important queue length is to measuring the performance of the array it is no surprise that there are a few strong positive correlations with regards to this metric.

Total Throughput

Total throughput is the measurement of IOPS on the storage array. As this number approaches its theoretical maximum the performance will decrease and queue length should decrease. That being said below in Figure 5.13 represents a bivariate analysis of throughput and queue length and we can see that it is somewhat bimodal. There is an extremely weak relationship between the two variables shown by $r^2 = 0.18$.

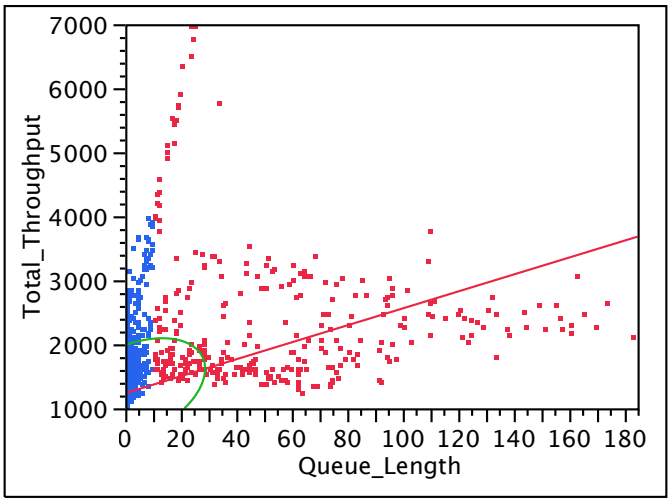


FIGURE 5.13: Bivariate: Total Throughput & Queue Length

Response Time

Response time is a critical metric in determining the user experience when using the storage array. When response time is low, the whole system will feel sluggish even if there are a plethora of cpu and memory resources available on the host part of the environment. The bivariate graph in Figure 5.14 details how strong the linear fit it with an r^2 value of 0.18.

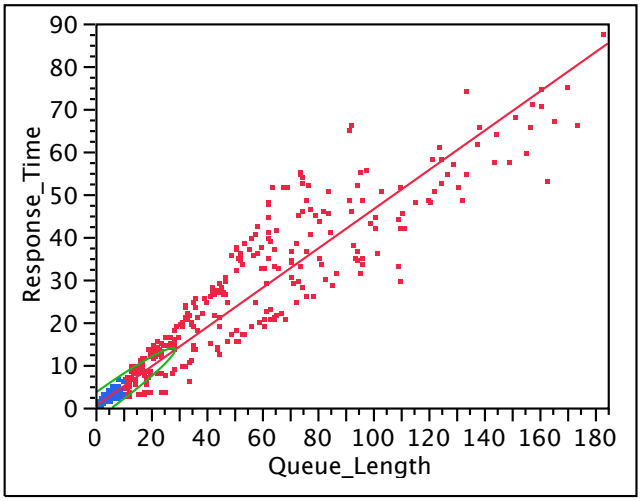


FIGURE 5.14: Bivariate: Response Time & Queue Length

Flush Ratio

Flush ratio is the number of times in a second that the cache on the service processor performs a write action to disk [10]. Writing to disk is much slower than working with cache, there are difference levels of flushing all with a difference impact on performance ranging from minimal to a large impact. A large value for flush ratio signifies a heavy back-end workload. The relationship between queue length and flush ratio is not linear but there is still a small positive quadratic relationship. The r^2 value of the relationship is 0.57 displayed in Figure 5.15.

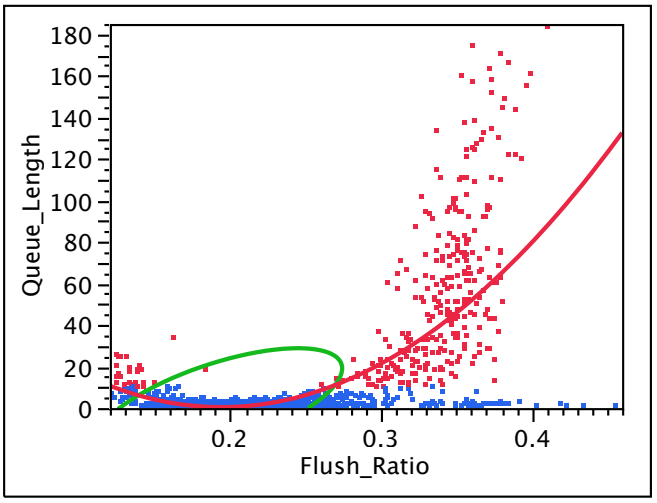


FIGURE 5.15: Bivariate: Flush Ratio & Queue Length

It is clear that there is a sweet spot where flush ratio must stay around in order to maintain optimal performance. Flush ratio is the rate at which data is taken from the cache on the service processors and written to disk.

5.3.7 LUN Performance

Take for example every LUN in the system, a great way to visualize the performance of each of these LUNs would be something along the lines of a bubble plot. JMP provides a great bubble plot platform to leverage for this analysis shown in Figure 5.16.

The bubble plot platform not only gives the admin a look into the queue length for every LUN simultaneously it is possible to see the throughput by how large each bubble is, the LUNs performance over time by the trailing transparent bubbles, and response time

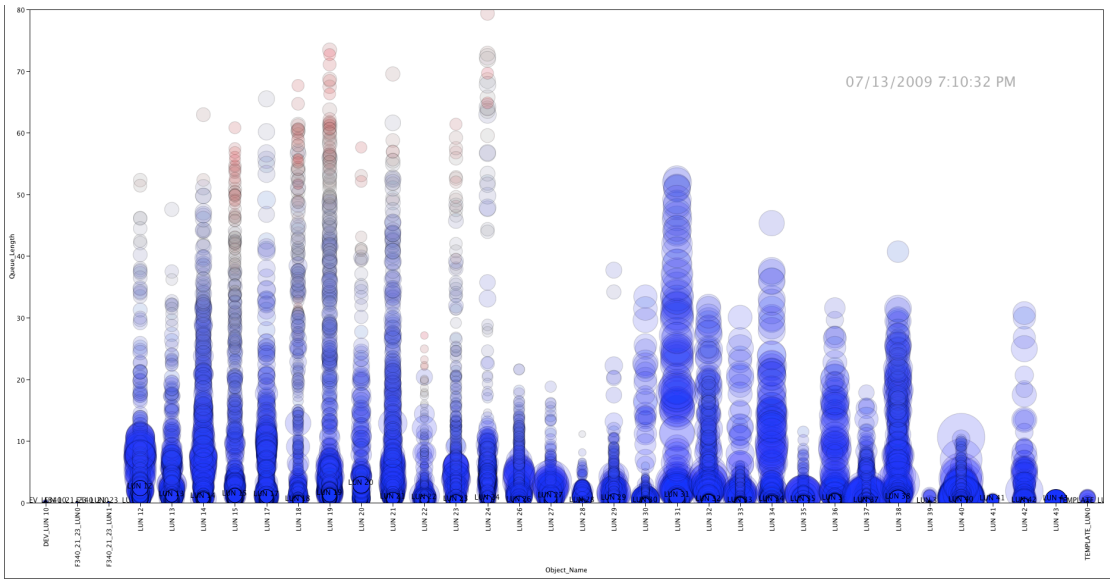


FIGURE 5.16: LUN Queue Length, Throughput, and Response Time

by what color each bubble is. This is a huge advantage because the admin instantly has a large picture of each LUN and can spot trouble spots quickly. For example, in Figure 5.16 it is clear that there are some LUNs which are performing very efficiently with a small queue length and very high throughput without impact to the response time. That being said there are about eight LUNs which are operating at a critical area due to the response time, small throughput, and/or extreme queue length. This can be resolved by moving virtual machines off of LUNs to alleviate the contention or to make the LUNs smaller.

5.3.8 Disk Performance

Another important aspect of overall storage performance is individual disk performance. Disks are combined into RAID groups which service LUNs. Aggregate disk performance provide storage arrays with the massive amount of performance within a shared system. Figure 5.17 shows the read and write performance in a bubble plot. Each bubble is colored from blue to red based on response time and the size is controlled by the queue length. That being said small dark blue bubbles are the best with regards to performance, big and red will usually point to poor performance from that individual disk. Keep in mind we are at the disk level and so a slow disk or two will not impact the array a great deal unless they reside within the same RAID group.

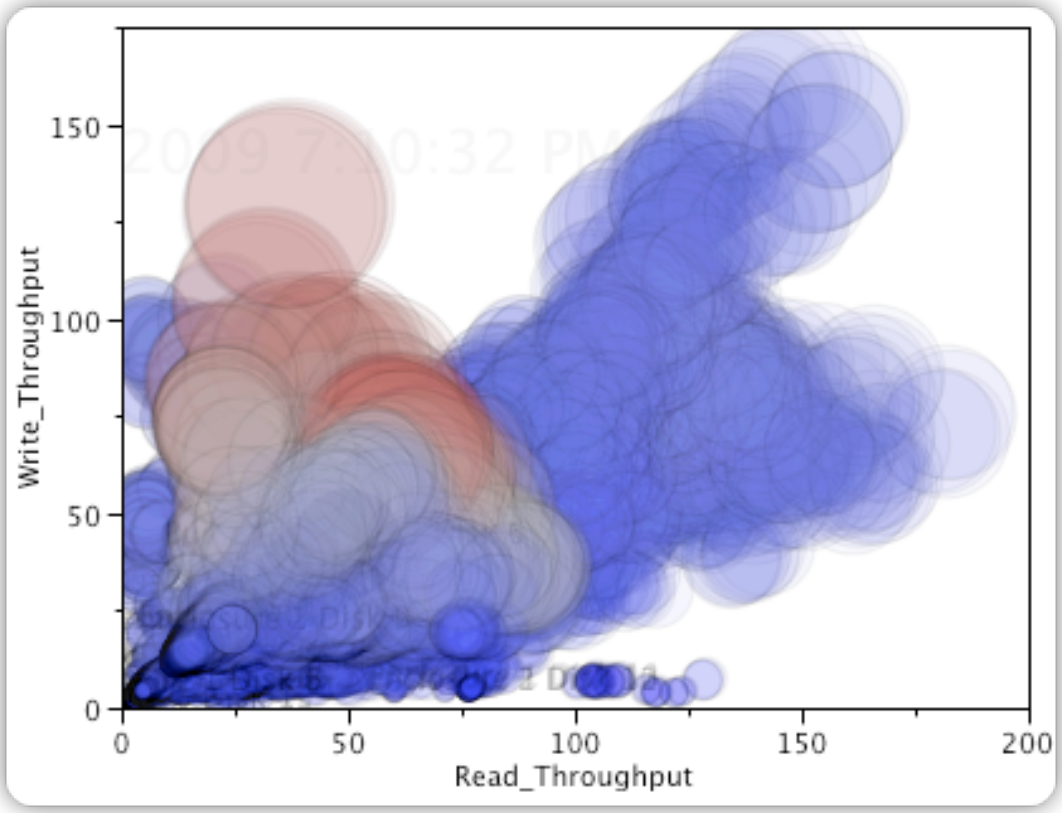


FIGURE 5.17: Individual Disk Bubble Plot

5.4 Conclusion

There is a plethora of information that can be collected regarding virtual environments. By collecting on the most important counters we can focus the analysis on what matters while still producing accurate results. Between the accessibility of the web application and the in-depth data visualization provided by JMP, a better understanding of the virtual environment is possible.

The web application and in-depth analysis provide a great sliding snapshot of the environments' health. Continually analyzing the environment and making improvements where necessary is imperative to maintaining a virtual environment and ensuring that it is operating at an optimal level. Without these tools it would be near impossible to view and compare the operation of each component of a virtual environment efficiently and draw any conclusions from that data. By having the data available in a relational database for just-in-time retrieval it is possible to recall and the data from any point in time and compare the environment now to that time.

6

Conclusion

6.1 Overview

Providing the ability to monitor a virtual environment in a visual manner to maintain and improve the performance of the system without increasing the amount of hardware in the system is extremely valuable. While performance can be measured in raw numbers it is not always cut and dry, performance of a virtual machine is dependent on many factors which are out of the virtual environment as well. It would be difficult to manage and monitor each function between the end-user and the virtual machine. It should be noted that if the virtual environments performance is monitored and reported at an accessible location the user is more apt to access the performance issues that may be seen are not attributed to the virtual environment but a different piece of the puzzle.

6.2 Web Application

In order to provide the accessible virtual environment monitoring application a web application was a perfect choice. It provides information to the end-user and admin irregardless to the platform they are using. A dynamic database-driven web application allowed for the rapid development of a functional application for use to monitor trends within a virtual environment. The reason for the rapid development was because the web application was able to leverage the two processes which were native to the storage

and server environment and glue them together in a cohesive web application without any modifications.

While the web application was extremely important to monitoring this virtual environment it did not provide enough detail at times. The application served its purpose, to provide a quick overview and single-host or storage array object detailed views. Other than those uses it fell short, this is when an application such a JMP can be used to provide a much more detailed view of each component due to its flexibility.

6.3 Host Analysis

While the storage array portion of the environment is extremely important there are few areas for improvements aside from load balancing LUNs and adding more disks. More on storage array performance in Section 5.3. Initially there can be a lot of interaction within a host environment to quickly improve the performance by implementing stricter guidelines for working within a virtual environment. These guidelines need to be similar to the same policy and procedures used for physical servers. For example policies regarding.

- Virtual Machine Creation
- Adding Storage
- Increasing Available Resources

By regulating these actions listed about, the performance of a virtual environment is more likely to stay constant and grow at a more manageable rate. These types of requirements are very easy to manage when it comes to physical servers because their purchase usually requires approval from a manager or a different third-party. These restrictions do not usually exist within a virtual environment so other options need to be developed.

On top of the basic guidelines that should be followed there are additional important actions to monitor and regulate specific to a virtual environment as opposed to a physical server environment. For example access control is extremely important when using a virtual environment. By limiting the number of users who have access and the number

of access methods they can use, it can greatly reduce the probability of a performance and/or security issue within the environment.

Since the host part of a virtual environment directly impacts the storage side of the environment, it is imperative to manage and monitor the use and actions performed on the user facing side of the environment. Some options that can be used to limit the available actions that can be performed within the host side of the environment are as follows.

- Access Reduction
  - User Restriction
  - Role Based Access Control
  - User Limitations
- Monitoring
  - Ongoing Auditing
  - Notification
  - Consolidation
  - Stagnant VM Policies

6.3.1 Access Reduction

Access reduction to the end user consists of three major parts, restricting users, creating roles for each user to limit their privilege scope within the system, and lastly enforcing stick user limits like quotas and number of VM limits on top of all that.

In the virtual environment used for this thesis there have been improvements to the authentication system and interface to it to take advantage the access restrictions stated. Some of the restrictions that were put in place were as follows:

- Leverage Active Directory
- Console Access via VNC
- Remove All Access via VI Client

- Disk Quotas via API
- Max VM "Slots" per User

The rate at which virtual machines were create after these improvements were made was decrease. THe most effective method to combat VM sprawl was the quotas that were implemented. This prevented users from using all of their slots with large drives.

That being said, users were still happy with the service provided, the restrictions put in place did not adversely effect the quality of the users experience. It was very important to ensure the end-users were satisfied with the interaction with the virtual environment as well as maintaining the performance. Between automated monitoring and the restrictions put in place, that goal is accomplished.

6.3.2 Monitoring

By combining a mixture of monitoring of a virtual environment with visualization of data and the ability to perform an in-depth analysis. These three requirements really enable a comprehensive way to analyze the virtual environment.

6.4 Storage Analysis

The storage array in particular collects a vast amount of data for analysis later on. It is up to the administrator to sift through this data and generate a report that is coherent and meaningful. Many counters are polled within the storage array, only a few of these counters actually need to be monitored on a regular basis. By limiting the overall amount of data the efficiency of the analysis is improved.

For example we have a perfectly sufficient weekly bubble plot shown in Figure 5.16. With this plot alone we can see contention within a LUN as well as good candidates to migrate virtual machines to. This LUN bubbled plot can be scripted in JMP in about just a few lines. A script that will produce the bubble plot is shown in Appendix G. By leveraging JMP scripting using the JSL[1] language built into the JMP software package, an admin can offer automated analysis not only via the web application but also through JMP.

[1] JMP Scripting Language

7

Recommendations

7.1 Future Work

While the suite of applications and scripts developed for this thesis are usable they can be extended in so many ways to improve the information gained as well as the efficiency. This project could be a great starting point for anyone who would like to monitor a virtual environment from 1 host to hundreds of hosts.

7.1.1 Collection Improvements

Data collection is an important aspect of the monitoring process for a virtual environment. Without an efficient way to collect and store data it would be impossible to manage the amount of data that can be collected. To make the storage of the information that is collected more efficient and manageable, a process to rollup the data based on certain intervals could be implemented.

There is a large amount of data collected every hour for a virtual environment, this amount of data cannot be effectively stored and represented via a simple web application or offline analysis tool. In order to decrease the amount of data without loosing too much resolution, we can rollup the data at select intervals. VMware already does this for their virtual center product. Since the most recent data is also the most important, we will ensure the resolution of that data is the highest. Then after a number of days pass

Days	Resolution
1	Full
2-7	1 Hour
8-30	6 Hours
31-90	1 Day
91-365	1 Week

TABLE 7.1: Data Rollup Policy

the data is still relevant to analyze trends in usage but down to the minute data is not necessary. That being said we can reduce the detail of the data by averaging each hour of data. Refer to Table 7.1 below for the data detail summarization.

7.1.2 Available Counters

VMware can collect data for many different counters, not just CPU, Memory, and Disk IO activity. These counters would provide an even greater detail about a virtual environment. That being said, this can only be implemented if a data rollup policy is implemented.

7.1.3 Realtime Analysis

Each system collects the data at the end of the day, this is fine if you do not need realtime analysis. The requirement for just-in-time analysis is increasing, by implementing a direct-to-database data collection system it could provide realtime analysis.

7.1.4 Flexible Visualization Code

The application used to display the information about each server could be improved to allow multiple servers on a single graph. Other options that could implement this graphing advancement could also be the ability to view performance of a cluster or resource pool. That would require more fields in the database to keep track of which hosts are in which resource pool and cluster they belong to.

7.1.5 Pure Ruby Implementation

The VMware Perl API toolkit is very memory and CPU hungry. The performance of the API is very poor and introduces a great deal of latency in the collection mechanism. Another options is to use the VMware SOAP API and implement the collection in pure

ruby to improve the speed as well as the ability to integrate it directly as a background task in the web application.

7.1.6 Notifications

One great feature for the web application would to perform a notification to the administrator(s) when a certain threshold is reached. By notifying the admin of a problem host(s) the admin then would be able to stay on top of any virtual machine migrations to different LUNs. Eventually after the reliability of the content of the notification it could be automated via the VMware API.

7.2 Conclusion

Considering projects are never completed, they just reach a complete milestone, there is still work that can be done to all of the parts of this thesis. For example both the web application and the data collection can be improved upon with regards to the efficiency and capabilities. By improving the efficiency of the web application more users are apt to use the application and rely on it. By open-sourcing the application and data collection code more developers will be able to provide input and submit updates to the program, this will ideally enhance the program and increase the adoption of the application.

A

Key Terms

A.1 Virtualization

VM Virtual Machine, a virtual disk on a host machine which encapsulates an operating system in a file usually stored on a high performance file-system.

SAN Storage Area Network - High performance network for drives shared between hosts in a virtual environment.

IOPS I/O Operations Per Second - IOPS are usually a better metric for VM performance than MB/sec.

A.2 Programming

ORM Object Relational Mapping - provides an abstraction layer between the web application and the database. Enables the web application to be database agnostic.

DRY Don't Repeat Yourself - Paradigm which dictates that code repetition should be eliminated if possible.

MVC Model View Controller - Paradigm which separates business logic, data display, and user interaction code.

API Application Programming Interface - Methods provided by the manufacturer used to communicate with the manufacturers' software/hardware.

B

Automation

B.1 Capistrano

This script below leverages the Capistrano gem which is an extensible ruby framework for automated server and deployment tasks. The script below allows me to push out the polling script to every ESX host in a cluster with one command. SSH keys are setup on each ESX host to allow a password-less connection in conjunction with ssh-agent.

```
task :installPoller, :user => "root", :roles => :vnap do
  upload "poll.sh", "/root/poll.sh", :via => :scp
  run "chmod +x /root/poll.sh"
  run "echo "03 0 * * * root /bin/bash /root/poll.sh" >> /etc/crontab"
end

task :disablePoller, :user => "root", :roles => :vnap do
  run "sed '$d' < /etc/crontab > /tmp/crontab"
  run "mv -f /tmp/crontab /etc/crontab"
end
```

C

Host Data Collection

C.1 ESX Host

```
#!/bin/bash

ESXCMD='/usr/bin/esxtop -s -b -d 180 -n 480 -c /root/.esxtop310rc'
LOCALDIR='/tmp'
REMOTEDIR='VNAP/'`hostname -s`'/'
REMOTESVR='calo-xperch.cisco.com'
REMOTEUSER='esx'
FILENAME=`hostname -s`'-'`date +%Y-%m-%d`'.csv'

$ESXCMD > $LOCALDIR/$FILENAME 2>/dev/null

scp $LOCALDIR/$FILENAME $REMOTEUSER@$REMOTESVR:$REMOTEDIR
rm $LOCALDIR/$FILENAME
```

C.2 esxtop configuration

The command `esxtop` was used to collect data from each esx server. The configuration file below determines the data that is collected from the command. A lowercase letter corresponds to the collection being turned off and uppercase is on. By disabling a large portion of unused data it optimized the data-set and improved the manageability of the data.

```
abcdefgh
abcdefghijklmno
AbcdEfghIjklm
AbcdefGhijk
abCdefghIjk
abcDefGhIjklm
5c
```

D

Data Summarization

D.1 Host Summarization

```
host_connection = Host.connection
@host_dates = host_connection.select_values
  'SELECT DISTINCT(DATE_FORMAT(timestamp, "%Y-%m-%d"))
  FROM hosts'

@dailyCPUAvg = []
@dailyMemAvg = []
@dailyDiskAvg = []

@host_dates.each do |date|
  cpu = []
  mem = []
  disk = []

  @cpuCounter = Host.find(:all, :conditions =>
    ["counterName = ? AND timestamp >= ? AND
    timestamp < ?", "cpu usagemhz", date.to_date,
    (date.to_date+1)])
```

```
@memCounter = Host.find(:all, :conditions =>
  ["counterName = ? AND timestamp >= ? AND
  timestamp < ?", "mem usage", date.to_date,
  (date.to_date+1)])

@diskCounter = Host.find(:all, :conditions =>
  ["counterName = ? AND timestamp >= ? AND
  timestamp < ?", "disk usage", date.to_date,
  (date.to_date+1)])

@cpuCounter.each do |c|
  cpu << c.value
end

cpu_avg = cpu.sum / cpu.length
hs = HostSummary.new(:timestamp => date.to_date,
  :counterName => @cpuCounter.first.counterName,
  :value => cpu_avg, :unit => @cpuCounter.first.unit)
hs.save!

@memCounter.each do |m|
  mem << m.value
end

mem_avg = mem.sum / mem.length
hs = HostSummary.new(:timestamp => date.to_date,
  :counterName => @memCounter.first.counterName,
  :value => mem_avg, :unit => @memCounter.first.unit)
hs.save!

@diskCounter.each do |d|
  disk << d.value
end
```

```
  disk_avg = disk.sum / disk.length
  hs = HostSummary.new(:timestamp => date.to_date,
    :counterName => @diskCounter.first.counterName,
    :value => disk_avg, :unit => @diskCounter.first.unit)
  hs.save!
end
```

D.2 Array Summarization

```
array_connection = StorageArray.connection
@array_dates = array_connection.select_values
  'SELECT DISTINCT(DATE_FORMAT(Poll_Time, "%Y-%m-%d"))
  FROM storage_arrays'

@array_dates.each do |date|
  puts "Summarizing Array Data for #{date}."
  @sps = StorageArray.find(:all, :conditions =>
    ["Object_Name LIKE ? AND Poll_Time >= ? AND
    Poll_Time < ?", "SP%", date.to_date, (date.to_date+1)])

spa_utilization             = []
spb_write_throughput        = []

@sps.each do |e|
  case e.Object_Name
  when "SP A" then
    spa_utilization << e.Utilization
  when "SP B" then
    spb_utilization << e.Utilization
  end
end

spa_util  = spa_utilization.sum / spa_utilization.length
```

```
spb_util  = spb_utilization.sum / spb_utilization.length

as = ArraySummary.new(:Object_Name => "SP A",
  :Poll_Time => date.to_date,
  :Utilization => spa_util,
  :Queue_Length => spa_queue,
  :Response_Time => spa_resp,
  :Total_Bandwidth => spa_tband,
  :Total_Throughput => spa_tthru,
  :Read_Bandwidth => spa_rband,
  :Read_Throughput => spa_rtru,
  :Write_Bandwidth => spa_wband,
  :Write_Throughput => spa_wthru)
as.save!

as = ArraySummary.new(:Object_Name => "SP B",
  :Poll_Time => date.to_date,
  :Utilization => spb_util,
  :Queue_Length => spb_queue,
  :Response_Time => spb_resp,
  :Total_Bandwidth => spb_tband,
  :Total_Throughput => spb_tthru,
  :Read_Bandwidth => spb_rband,
  :Read_Throughput => spb_rtru,
  :Write_Bandwidth => spb_wband,
  :Write_Throughput => spb_wthru)
as.save!
```

E

ESX Host Data Collection

E.1 VMware API Perl Script

```
#!/usr/bin/env perl
use strict;
use warnings;
use DBI;
use DBD::mysql;
use VMware::VIRuntime;
$Util::script_version = "1.0";

sub get_perf;

my $server = 'SERVER.DOMAIN';
my $username = 'USERNAME';
my $password = 'PASSWORD';

my $database = "DATABASE";
my $mysqluser = "USERNAME";
my $mysqlpassword = "PASSWORD";
```

```
Util::connect("https://$server/sdk/vimService",$username,$password);
my $dbh = DBI->connect("DBI:mysql:$database","$mysqluser"
  ,"$mysqlpassword") or die "Couldn't connect to database: "
  .DBI->errstr;
get_perf();
Util::disconnect();
$dbh->disconnect;

sub get_perf() {
  my $all_counters;
  my $hosts = Vim::find_entity_views(view_type => "HostSystem");
  my $perfmgr_view = Vim::get_view(mo_ref =>
    Vim::get_service_content()->perfManager);
  my $perfCounterInfo = $perfmgr_view->perfCounter;

  foreach (@$perfCounterInfo) {
    if ($_->rollupType->val =~/average/) {
      my $key = $_->key;
      my $group_info = $_->groupInfo;
      my $name_info = $_->nameInfo;

      if ($group_info->key eq 'disk' && ($name_info->key eq 'usage')) {
        $all_counters->{ $key } = $_;
      }
      elsif ($group_info->key eq 'mem' && ($name_info->key eq
       'usage')) {
        $all_counters->{ $key } = $_;
      }
      elsif ($group_info->key eq 'cpu' && ($name_info->key eq
        'usagemhz')) {
        $all_counters->{ $key } = $_;
      }
    }
  }
```

```
my @filtered_list;
foreach my $host (@$hosts) {
  my $entity = $host;
  my $counters = $all_counters;
  my $perf_metric_ids;

  $perf_metric_ids = $perfmgr_view->
    QueryAvailablePerfMetric(entity => $entity);

  if (scalar(@filtered_list) < 1) {
    foreach (@$perf_metric_ids) {
      if (exists $counters->{$_->counterId}) {
          push @filtered_list, $_;
      }
    }
  }
  $perf_metric_ids = \@filtered_list;

  my $intervals = get_available_intervals(perfmgr_view =>
    $perfmgr_view, host => $host);

  (my $sec,my $min,my $hour,my $mday,my $mon,my $year,my $wday,
     my $yday,my $isdst)=localtime(time);

  my $startDate = sprintf "%4d-%02d-%02dT00:00:00"
    ,$year+1900,$mon+1,$mday;

  my $endDate = sprintf "%4d-%02d-%02dT23:30:00"
    ,$year+1900,$mon+1,$mday;

  my $perf_query_spec = PerfQuerySpec->new(entity => $entity,
    metricId => $perf_metric_ids, format => 'csv',
    startTime => $startDate, endTime => $endDate);
```

```
my $perf_data = $perfmgr_view->QueryPerf
  (querySpec => $perf_query_spec);

foreach (@$perf_data) {
  my $time_stamps = $_->sampleInfoCSV;
  my $values = $_->value;
  my @ts = split(/(?<![0-9]),/,$time_stamps);

  foreach my $val (@$values) {
    my @valuelist = split /[,]/, $val->value;
    my $ts_ctr = 0;
    my $factor = 1;

    my $currentCounter = $all_counters->{$val->id->counterId};

    if ($currentCounter->unitInfo->label eq 'Percent') {
        $factor = 100;
    }

    foreach my $v (@valuelist) {
      my $index;
      my $length;
      my $time = $ts[$ts_ctr];

      $index = index($time,",");
      $time = substr($time,($index+1));
      $time =~ s/T/ /;
      $length = length($time);
      $time = substr($time, 0, ($length-1));

      my $sth = $dbh->prepare("INSERT INTO hosts
        (timestamp,hostname,counterName,value,unit)
        VALUES (\"".$time."\",\"".$host->name."\",\""
```

```
            .$currentCounter->groupInfo->key." "
            .$currentCounter->nameInfo->key."\",\""
            .$v/$factor."\",\"".$currentCounter->unitInfo->key
            ."\")") or die "Couldn't prepare statement: "
            . $dbh->errstr;

          $sth->execute() or die "Couldn't execute statement: "
            .$sth->errstr;
          $ts_ctr ++;
        }
      }
    }
  }
}

sub get_available_intervals {
  my %args = @_;
  my $perfmgr_view = $args{perfmgr_view};
  my $entity = $args{host};
  my $historical_intervals = $perfmgr_view->historicalInterval;
  my $provider_summary = $perfmgr_view->QueryPerfProviderSummary
    (entity => $entity);

  my @intervals;
  if ($provider_summary->refreshRate) {
    push @intervals, $provider_summary->refreshRate;
  }
  foreach (@$historical_intervals) {
    push @intervals, $_->samplingPeriod;
  }
  return \@intervals;
}
```

F

Web Application Database Schema

F.1 Schema Design

Field	Type
Object_Name	varchar(255)
Poll_Time	date
Utilization	float
Queue_Length	float
Response_Time	float
Total_Bandwidth	float
Total_Throughput	float
Read_Bandwidth	float
Read_Throughput	float
Write_Bandwidth	float
Write_Throughput	float

Field	Type
timestamp	datetime
counterName	varchar(255)
value	float
unit	varchar(255)

Table F.1: veViz Summary Schema

Field	Type
Object_Name	varchar(255)
Poll_Time	datetime
Owner_Array_Name	varchar(255)
Current_Owner	varchar(255)
Utilization	float
Queue_Length	float
Response_Time	float
Total_Bandwidth	float
Total_Throughput	float
Read_Bandwidth	float
Read_Size	float
Read_Throughput	float
Write_Bandwidth	float
Write_Size	float
Write_Throughput	float

Field	Type
timestamp	datetime
hostname	varchar(255)
counterName	varchar(255)
value	float
unit	varchar(255)

Table F.2: veViz Master Schema

G

JMP Scripts

G.1 LUN Bubble Plot

```
Bubble Plot(
  X( :Object_Name ),
  Y( :Queue_Length ),
  Sizes( :Total_Throughput ),
  Time( :Poll_Time ),
  Coloring( :Response_Time ),
  ID( :Object_Name ),
  Speed( 220 ),
  Time Index( 1953 ),
  Trail Bubbles( 1 ),
  All Labels( 0 ),
  Legend( 1 ),
  SendToReport(
    Dispatch( {}, "2", ScaleBox, {Min( 0 ), Minor Ticks( 0 )} ),
    Dispatch( {}, "Bubble Plot", FrameBox, Frame Size( 932, 272 ) )
  )
)
```

H

Storage Array Collection Scripts

H.1 Storage Data Collection Script

```
#!/usr/bin/env ruby
require 'rubygems'
require 'net/ping'
require 'net/scp'
require 'net/ssh'

class Storage
  attr_accessor :downloads, :last_download, :files, :convert_dir
  def initialize(ip)
    @array_ip = ip
    @last_download = ""
    @files = []
    @downloads = "#{Dir.pwd}/downloads"
    @convert_dir = "#{Dir.pwd}/convert"
  end
  def ping?
    begin
      return Net::Ping::ICMP.new(@array_ip).ping
```

```
    rescue Exception => e
      puts "ERROR: #{e}"
      exit
    end
  end
  def data_source?
    return File.exists?("data")
  end
  def data_dir?
    if File.exists?(@downloads)
      return File.directory?(@downloads)
    else
      return false
    end
  end
  def download
    Dir.chdir(@downloads)
    @files.each do |f|
      puts "Downloading #{f}"
      system "yes | /opt/Navisphere/bin/naviseccli
        -h #{@array_ip} analyzer -archive -file #{f} -o"
    end
  end
  def convert
    Dir.chdir(@convert_dir)
    @files.each do |f|
      puts "Converting #{f} to CSV"
      system "yes | /opt/Navisphere/bin/naviseccli
        -h #{@array_ip} analyzer -archivedump -data
        #{@downloads}/#{f} -out #{f}.csv"
    end
  end

  def cleanup
```

```
    Dir.chdir(@convert_dir)
    csv_files = Dir.entries(@convert_dir)
    csv_files.each do |f|
      if File.file?(f)
        Net::SSH.start("calo-veviz", "rvalente",
            :password => "command7") do |ssh|
          ssh.scp.upload! "#{@convert_dir}/#{f}", "/tmp/emc/"
          puts "SCP Successful"
          File.delete(f)
        end
      end
    end
  end

  def refresh
    start_date = Time.now
    puts "Syncing Data Source with Array..."
    system "/opt/Navisphere/bin/naviseccli -h
      #{@array_ip} analyzer -archive -list > data"
    if start_date > File.mtime("data")
      return false
    else
      return true
    end
  end

end

print "Enter Array IP: "
array = gets.chomp!

s = Storage.new(array)
if s.ping?
  puts "Array can be reached."
```

```
else
  puts "Array cannot be reached, exiting."
  exit
end

if s.data_source?
  @data = File.new("data", "r")
else
  puts "Missing Data Source File, Updating..."
end

if s.refresh
  puts "Successfully Updated data source"
else
  puts "Data Source Update Failed"
  exit
end

if s.data_dir?
  previous_downloads = Dir.entries(s.downloads)
  previous_downloads.sort!
  s.last_download = previous_downloads.last
  puts "Last Downloaded File: #{s.last_download}"
else
  puts "No Data Dir!"
  exit
end

counter = 0
@files = []
@collect = false

@data.each_with_index do |line,index|
  if counter == 1
```

```
    counter = counter + 1
  end

  line = line.squeeze(" ")
  line_sp = line.split
  data = line_sp[4]

  if data.index(s.last_download)
    counter = counter + 1
    @collect = true
  end
  if @collect && counter == 2
    s.files << data
  end

end

@data.close
s.download
s.convert
s.cleanup
```

Bibliography

[1] Tobias Oetiker. Multi router traffic grapher, 2009. URL `http://oss.oetiker.ch/mrtg/`.

[2] Tobias Oetiker. Rrdtool, 2009. URL `http://oss.oetiker.ch/rrdtool/`.

[3] Darrell Reimer, Arun Thomas, Glenn Ammons, Todd Mummert, Bowen Alpern, and Vasanth Bala. *Opening Black Boxes: Using Semantic Information to Combat Virtual Machine Image Sprawl*. 2008.

[4] MySQL Development Team. Mysql relational database, 2009. URL `http://dev.mysql.com/`.

[5] amCharts. amcharts - xml graphing solution, 2009. URL `http://www.amcharts.com/`.

[6] John Sall, Lee Creighton, and Ann Lehman. *JMP Start Statistics: A Guide to Statistics and Data Analysis Using Jmp*. SAS Institute Inc, Cary, NC, 2007.

[7] VMware. Vi performance monitoring, 2009. URL `http://download3.vmware.com/media/partners/techexchange/vi05.html`.

[8] Ann Lehman Norm O'Rouke Larry Hatcher Edward J. Stepenski. *JMP for Basic Univariate and Multivariate Statistics: A Step-by-Step Guide*. SAS Institute Inc, Cary, NC, 2005.

[9] SAS. *JMP 8 Statistics and Graphics Guide Volumes 1 and 2*. SAS Institute Inc, Cary, NC, 2008.

[10] EMC. Navisphere analyzer: A case study. 2001.